

**Evaluation of Guidance for Tools Used to Develop
Safety-Related Digital Instrumentation and Control Software
for Nuclear Power Plants**

Task 2 Report: Analysis of the State of Practice

Prepared by:
James Servatius
Stephen Alexander
Terry Gitnick

Reviewed by:
William Arcieri

Energy and Space Division
Information Systems Laboratories, Inc.
11140 Rockville Pike, Suite 650
Rockville, MD 20852

August 2014

Prepared for:

U.S. Nuclear Regulatory Commission
Office of Nuclear Regulatory Research
NRC Contract No: NRC-HQ-13-C-04-0004
ISL Project No. 2310

Intentionally Blank

Executive Summary

Application of software tools in new reactor design, development, and existing reactor upgrade and operations has been increasing in recent years. In general, the use of software tools is more efficient than the traditional development process and may result in fewer faults than a manual engineering process if the tools are well designed, carefully developed, rigorously tested, and appropriately used. However, undetected faults in the automated tools or tool-assisted engineering activities may pose serious risks to nuclear safety. Having a good process and consistent method to assess the safety of software tool use in nuclear safety systems is important to all stakeholders in the nuclear industry from equipment vendors, utility licensees, and government regulatory organizations. However, there is currently no specific and detailed standard and commonly accepted practice in the US nuclear industry which addresses software tool qualification or certification.

NRC and Information Systems Laboratories, Inc. (ISL), the contractor, launched a project focused on the safety assessment of software tool use in nuclear safety systems. This project reviewed more than 150 industrial standards, government positions, and regulatory guidance, which include nuclear, aerospace, civil aviation, railway, and automotive industries, different standard organizations, such as IEEE, IEC, ISO, EPRI, IAEA, and RTCA, and many government organizations, such as NASA, FAA, NRC, AECL, and NIST. The purpose is to learn from a broad spectrum of expertise and extract the most important and relevant ideas to establish the technical bases for potential regulatory guidance for the safety assessment of software tool use in the development, operation, and maintenance of US nuclear safety systems.

Task 1 of this project was to survey the field of available literature of various industries that make use of safety-critical computer-based systems and software. ISL found a plethora of information that is applicable to software, directly or indirectly, some that is relevant to software tools, again, directly or indirectly, and some that, while not written for software tools, could be adaptable and useful in the safety assessment of software tools. This information included regulations or regulatory requirements, regulatory guidance, regulatory information, industry guidance, and industry information. In addition, ISL presented the findings of its research and discussed the various types of software tools that are used in the various life cycle processes of safety-critical software development, verification and validation, testing, installation and integration, operation, and maintenance. From this information, ISL extracted a set of baselines that have relevance to several aspects of software tool evaluation by users and regulatory oversight of the evaluation processes.

In this phase (Task 2) of the project, ISL performed detailed analyses of the software-related information, compared and extracted the requirements and guidelines from those documents, and generated draft baselines of possible NRC requirements and guidelines for software tool evaluation. ISL also performed detailed analyses of software and software tool classification methodologies that determine the appropriate level of rigor to be applied in the process of verifying the tools' suitability, quality and reliability through what other industries call tool qualification. This could form the basis for acceptance of tools by licensees or applicants

developed under nuclear QA programs that meet 10 CFR Part 50, Appendix B, as well as licensee or applicant acceptance of what is expected to be the majority of tools used, which were not developed under such a nuclear QA program through the process of commercial-grade dedication. These analyses also resulted in information that can be used by the NRC in developing regulatory guidance for software tools as well as internal guidance for the review of applications for NRC approval of software tools to be used in the development and testing of software associated with systems important to safety in NRC-licensed facilities.

This report comprises six major sections. Section 1 of this Task 2 report is an introduction, background discussion, and overview. Section 2 presents the results of ISL's analyses of the documents from the various sources, including recommendations on enhancement of existing guidance and possible new guidance. Section 3 presents a comparison of the information analyzed among the industries surveyed thus far, including, nuclear (domestic and international), aerospace, civil aviation, railroad, and automotive. Section 4 discusses software tool hazard analysis. Section 5 discusses the baselines that have been distilled and refined from the analyses. Section 6 presents a summary of ISL's findings and observations, and Section 7 gives references. Finally, this task organized and assessed the information to enable the derivation from it in Task 3 of elements with which to develop the technical bases for possible NRC regulatory requirements and guidance and enhancing existing requirements and guidance on acceptable means of or processes for licensee or applicant acceptance of software tools to be used with important-to-safety software, and also guidance on the processes for NRC review and approval of applications for the use of such software tools. ISL expects to provide information that will be useful to the NRC in both a deterministic and risk-informed approach to regulatory oversight in this area. It is also expected that the NRC will be able to make use of both the review and approval process information and the detailed licensee or applicant acceptance information in a two-pronged approach to regulatory oversight in this area, which could comprise oversight of the software tool acceptance activities by licensees and applicants in conjunction with independent NRC evaluation of selected tools.

Table of Contents

Executive Summary	iii
Abbreviations and Acronyms.....	xiii
Glossary.....	xv
1 Introduction	1
1.1 Background.....	1
1.2 Overview.....	3
2 Document Analysis	5
2.1 Aerospace Industry Analysis.....	5
2.1.1 Document Analysis	5
2.1.1.1 <i>NASA Procedural Requirement 7150.2A</i>	5
2.1.1.2 <i>NASA-STD-7009</i>	10
2.1.1.3 <i>NASA-STD-8739.8</i>	10
2.1.1.4 <i>NASA-STD-8719.13C</i>	12
2.1.1.5 <i>NASA-GB-8719.13</i>	14
2.1.1.6 <i>NASA-HDBK-8739.23</i>	18
2.1.2 Summary	19
2.2 Civil Aviation Industry Analysis.....	22
2.2.1 Document Analysis	22
2.2.1.1 <i>FAA-STD-026A</i>	22
2.2.1.2 <i>RTCA DO-178C</i>	23
2.2.1.3 <i>RTCA DO-330</i>	31
2.2.1.4 <i>FAA Advisory Circular 20-115C</i>	38
2.2.1.5 <i>FAA Order 8110.49</i>	39
2.2.2 Summary	45
2.3 Nuclear Regulatory Commission Analysis.....	50
2.3.1 Document Analysis	51
2.3.1.1 <i>Regulatory Guide 1.152</i>	51
2.3.1.2 <i>Regulatory Guide 1.168</i>	54
2.3.1.3 <i>Regulatory Guide 1.169</i>	56
2.3.1.4 <i>Regulatory Guide 1.170</i>	56
2.3.1.5 <i>Regulatory Guide 1.171</i>	58

2.3.1.6	<i>Regulatory Guide 1.172</i>	59
2.3.1.7	<i>Regulatory Guide 1.173</i>	61
2.3.1.8	<i>Interim Staff Guidance (ISG) 06</i>	61
2.3.1.9	<i>NUREG-0800 Branch Technical Position 7-14</i>	65
2.3.1.10	<i>NUREG-0800 Appendix 7.1-D</i>	69
2.3.1.11	<i>NUREG/CR-6263</i>	71
2.3.1.12	<i>NUREG/CR-6421</i>	72
2.3.2	Summary	73
2.4	Institute of Electrical and Electronics Engineers Analysis.....	80
2.4.1	Document Analysis	80
2.4.1.1	<i>IEEE 7-4.3.2</i>	80
2.4.1.2	<i>IEEE 603</i>	83
2.4.1.3	<i>IEEE 1012</i>	85
2.4.1.4	<i>IEEE 14102 and 1462</i>	87
2.4.2	Summary	87
2.5	International Electrotechnical Commission Analysis.....	90
2.5.1	Document Analysis	91
2.5.1.1	<i>IEC 61508</i>	92
2.5.1.1.1	<i>IEC 61508-0</i>	92
2.5.1.1.2	<i>IEC 61508-1</i>	94
2.5.1.1.3	<i>IEC 61508-2</i>	99
2.5.1.1.4	<i>IEC 61508-3</i>	102
2.5.1.1.5	<i>IEC 61508-4</i>	107
2.5.1.2	<i>IEC 61226</i>	107
2.5.1.3	<i>IEC 60880</i>	109
2.5.1.4	<i>IEC 62138</i>	115
2.5.2	Summary	117
2.6	Automotive Industry Analysis	122
2.6.1	Document Analysis	122
2.6.1.1	<i>ISO 26262</i>	122
2.6.1.1.1	<i>ISO 26262-6</i>	124
2.6.1.1.2	<i>ISO 26262-8</i>	126
2.6.2	Summary	131

2.7	Railway Industry Analysis	134
2.7.1	Document Analysis	135
2.7.1.1	<i>BS EN 50128</i>	135
2.7.2	Summary	141
2.8	International Atomic Energy Agency Analysis	144
2.8.1	Document Analysis	145
2.8.1.1	<i>IAEA NS-G-1.1</i>	145
2.8.1.2	<i>IAEA DS-431</i>	151
2.8.2	Summary	155
2.9	Electric Power Research Institute Analysis	159
2.9.1	Document Analysis	160
2.9.1.1	<i>EPRI NP-5652</i>	160
2.9.1.2	<i>EPRI TR-102260</i>	165
2.9.1.3	<i>EPRI TR-106439</i>	168
2.9.1.4	<i>EPRI TR-107339</i>	170
2.9.1.5	<i>EPRI TR-1025243</i>	175
2.9.2	Summary	179
2.10	National Institute of Standards and Technology Analysis	185
2.10.1	Document Analysis	186
2.10.1.1	<i>NIST SP 500-234</i>	186
2.10.2	Summary	187
2.11	Atomic Energy of Canada Limited Analysis.....	188
2.11.1	Document Analysis	188
2.11.1.1	<i>AECL CE-1001-STD</i>	188
2.11.2	Summary	190
3	Industry/Organization Comparisons	193
3.1	Aerospace Industry	193
3.2	Civil Aviation Industry.....	194
3.3	Nuclear Regulatory Commission.....	196
3.4	Institute of Electrical and Electronics Engineers.....	198
3.5	International Electrotechnical Commission.....	200
3.6	Automotive Industry	201
3.7	Railway Industry	202

3.8	International Atomic Energy Agency	203
3.9	Electric Power Research Institute	204
3.10	National Institute of Standards and Technology	204
3.11	Atomic Energy of Canada Limited	205
4	Software Tool Hazard Assessment	207
4.1	Management Support Tools	208
4.2	Implementation Tools	208
4.3	Modeling Tools	210
4.4	Construction Tools	211
4.5	Maintenance Tools	212
4.6	Documentation Tools	212
4.7	Configuration Management Tools	212
4.8	Quality Assurance Tools	213
4.9	Verification and Validation Tools	213
4.10	Conclusions	214
5	Baselines	215
5.1	Baseline Additions	215
5.1.1	Baseline Requirements Additions	215
5.1.1.1	<i>Software and Software Tool Classifications</i>	215
5.1.1.2	<i>Software Tool Planning</i>	216
5.1.1.3	<i>Software Tool Documentation</i>	217
5.1.1.4	<i>Software Tool Selection and Use</i>	219
5.1.1.5	<i>Software Tool Development, Qualification, and Dedication</i>	220
5.1.1.6	<i>Software Tool Quality Assurance</i>	222
5.1.1.7	<i>Software Tool Training</i>	223
5.1.1.8	<i>Software Tool Configuration Management</i>	224
5.1.1.9	<i>Software Tool Review and Approval</i>	224
5.1.2	Baseline Guidance Additions	225
5.1.2.1	<i>Software and Software Tool Classifications</i>	225
5.1.2.2	<i>Software Tool Planning</i>	225
5.1.2.3	<i>Software Tool Documentation</i>	225
5.1.2.4	<i>Software Tool Selection and Use</i>	226
5.1.2.5	<i>Software Tool Development, Qualification, and Dedication</i>	228

5.1.2.6	<i>Software Tool Quality Assurance</i>	230
5.1.2.7	<i>Software Tool Training</i>	230
5.1.2.8	<i>Software Tool Configuration Management</i>	230
5.1.2.9	<i>Software Tool Review and Approval</i>	231
5.2	Baseline Deletions	231
5.2.1	Baseline Requirement Deletions.....	231
5.2.2	Baseline Guidance Deletions	232
5.3	Final Baselines	234
5.3.1	Software and Software Tool Classifications	234
5.3.2	Software Tool Planning.....	239
5.3.3	Software Tool Documentation.....	242
5.3.4	Software Tool Selection and Use.....	247
5.3.5	Software Tool Development, Qualification, and Dedication.....	253
5.3.6	Software Tool Quality Assurance.....	264
5.3.7	Software Tool Training.....	266
5.3.8	Software Tool Configuration Management.....	266
5.3.9	Software Tool Review and Approval	269
6	Summary and Conclusions	273
6.1	Aerospace Industry.....	274
6.2	Civil Aviation Industry.....	274
6.3	Nuclear Regulatory Commission.....	276
6.4	Institute of Electrical and Electronic Engineers.....	279
6.5	International Electrotechnical Commission.....	279
6.6	Automobile Industry	280
6.7	Railway Industry	281
6.8	International Atomic Energy Agency	282
6.9	Electric Power Research Institute	283
6.10	National Institute of Standards and Technology	285
6.11	Atomic Energy of Canada, Ltd	286
7	References	287
7.1	Aerospace Industry.....	287
7.2	Civil Aviation Industry.....	287
7.3	Radio Technical Commission for Aeronautics.....	287

7.4 Nuclear Regulatory Commission.....288

7.5 National Standards289

7.6 International Standards.....291

7.7 Electric Power Research Institute293

7.8 National Institute of Standards294

7.9 Miscellaneous294

Tables

Table 2.1 – Tool Qualification Level Determination	31
Table 2.2 –Target Failure Measure for Low-Demand Mode of Operation	96
Table 2.3 –Target Failure Measure for High-Demand or Continuous Mode of Operation	96
Table 2.4 – Determination of the Tool Confidence Level (TCL)	128
Table 2.5 – Qualification of Software Tools Classified TCL3	128
Table 2.6 – Qualification of Software Tools Classified TCL2	129
Table 5.1 – Determination of the Tool Confidence Level (TCL)	216
Table 5.2 – Qualification of Software Tools Classified TCL3	220
Table 5.3 – Qualification of Software Tools Classified TCL2	221
Table 5.4 – Determination of the Tool Confidence Level (TCL)	236
Table 5.5 – Tool Qualification Level Determination	239
Table 5.6 – Qualification of Software Tools Classified TCL3	261
Table 5.7 – Qualification of Software Tools Classified TCL2	262
Table 5.8 – SCM Process Objective Control Categories	268

Intentionally Blank

Abbreviations and Acronyms

The following acronyms and abbreviations are used frequently throughout this report. The report convention is to define the acronym the first time it is used in each major section of the report (e.g., Section 1, 2, etc.) and to define all acronyms in the following list.

AC	FAA Advisory Circular
ACG	Automatic Code Generation
AECL	Atomic Energy of Canada, Ltd
ANSI	American National Standards Institute
ASIC	Application Specific Integrated Circuit
ASIL	Automotive Safety Integrity Level
BTP	Branch Technical Position
CANDU®	Canada Deuterium-Uranium
CASE	Computer-Aided Software Engineering
CAST	Certification Authorities Software Team
CCF	Common-Cause Failures
CFR	Code of Federal Regulations
CGI	Commercial-Grade Item
CM	Configuration Management
COTS	Commercial off-the-Shelf
CSL	Computer Systems Laboratory
DBE	Design-Basis Event
DI&C	Digital Instrumentation and Control
DICWG	Digital I&C Working Group (of the MDEP)
DID	Data Item Descriptions
DO	RTCA Document
DOT	U.S. Department of Transportation
EIA	Electronic Industries Alliance
EPRI	Electric Power Research Institute
EUC	Equipment Under Control
FAA	Federal Aviation Administration
FPGA	Field-Programmable Gate Array
GB	Guidebook (NASA)
GL	NRC Generic Letter
GOTS	Government off-the-Shelf
I&C	Instrumentation and Control
IAEA	International Atomic Energy Agency
ICE	In-circuit Emulator
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers

ISG	Interim Staff Guidance
ISL	Information Systems Laboratories, Inc.
ISO	International Organization for Standardization
IT	Information Technology
MDEP	Multinational Design Evaluation Program (of the NEA)
MOTS	Modified off-the-Shelf
NASA	National Aeronautics and Space Administration
NCIG	Nuclear Construction Issues Group (of EPRI)
NEA	Nuclear Energy Agency (of the OECD)
NIST	National Institute of Standards and Technology
NP	EPRI Notification Paper
NPR	NASA Procedural Requirements
NRC	U.S. Nuclear Regulatory Commission
NUREG	U.S. Nuclear Regulatory Publication
OECD	Organization for Economic Cooperation and Development
PAM	Post-Accident Monitoring
PDD	Programmable Digital Device
PLC	Programmable Logic Controller
QA	Quality Assurance
RG	Regulatory Guide (NRC)
RISC	Risk-informed Safety Class (RISC)
RTCA	Radio Technical Commission for Aeronautics
SCM	Software Configuration Management
SIL	Safety Integrity Level (IEC)
SIL	Software Integrity Level (IEEE)
SP	Special Publication
SQA	Software Quality Assurance
SRP	Standard Review Plan
SSCA	Software Safety Criticality Assessment
SSCs	Structures, Systems, and Components
STD	Standard
TQL	Tool Qualification Level
TR	Topical or Technical Report
UMS	User-Modifiable Software
V&V	Verification and Validation

Glossary

Accident: An unplanned event or series of events that result in death, injury, illness, environmental damage, or damage to, or loss of, equipment or property. {RG 1.173, Revision 1}

Activity: Tasks that provide a means of meeting the objectives. {RTCA DO-178C}

Application-specific integrated circuit (ASIC): Integrated circuit designed and manufactured for specific function, where its functionality is defined by the product developer. {IEC 61508-4, Section 3.2.15}

Assessment: The process of analysis to determine whether software, which may include process, documentation, system, subsystem hardware and/or software components, meets the specified requirements and to form a judgment as to whether the software is fit for its intended purpose. Safety assessment is focused on but not limited to the safety properties of a system. {BS EN 50128:2011, Section 3.1.1}

Assurability: Ability to confirm the certainties of correctness of a claim, based on evidence and reasoning.

Average frequency of a dangerous failure per hour: Average frequency of a dangerous failure of a safety-related system to perform the specified safety function over a given period of time. {IEC 61508-4, Section 3.6.19}

Average probability of dangerous failure on demand: The mean unavailability of a safety-related system to perform the specified safety function when a demand occurs from the equipment under control (EUC) or EUC control system. {IEC 61508-4, Section 3.6.18}

Basic component: A safety-related structure, system, or component as defined below. {10 CFR 21.3}

- (1) (i) When applied to nuclear power plants licensed under 10 CFR Part 50 or Part 52, basic component means a structure, system, or component, or part thereof that affects its safety function necessary to assure:
 - (A) the integrity of the reactor coolant pressure boundary;
 - (B) the capability to shut down the reactor and maintain it in a safe shutdown condition; or
 - (C) the capability to prevent or mitigate the consequences of accidents which could result in potential offsite exposures comparable to those referred to in § 50.34(a)(1), § 50.67(b)(2), or § 100.11, as applicable.
- (ii) Basic components are items designed and manufactured under a quality assurance (QA) program complying with 10 CFR Part 50, Appendix B, or

commercial-grade items (CGIs) which have successfully completed the dedication process.

- (2) Basic component applicability to standard design certifications under 10 CFR Part 52, Subpart C.
- (3) When applied to other facilities and other activities licensed under 10 CFR parts 30, 40, 50 (other than nuclear power plants), 60, 61, 63, 70, 71, or 72 of this chapter, basic component means a structure, system, or component, or part thereof, that affects their safety function, that is directly procured by the licensee of a facility or activity subject to the regulations in this part and in which a defect or failure to comply with any applicable regulation in this chapter, order, or license issued by the Commission could create a substantial safety hazard.
- (4) In all cases, basic component includes safety-related design, analysis, inspection, testing, fabrication, replacement of parts, or consulting services that are associated with the component hardware, design certification, design approval, or information in support of an early site permit application under part 52 of this chapter, whether these services are performed by the component supplier or others.

Certification: The legal recognition by the certification authority that a product, service, organization, or person complies with the requirements. Such certification comprises the activity of technically checking the product, service, organization, or person and the formal recognition of compliance with the applicable requirements by issue of a certificate, license, approval, or other documents as required by national laws and procedures. {RTCA DO-178C}

Certification authority: The organization or person responsible within the state, country, or other relevant body, concerned with the certification or approval of a product in accordance with requirements. {RTCA DO-178C}

Class 1E: The safety classification of the electric equipment and systems that are essential to emergency reactor shutdown, containment isolation, reactor core cooling, and containment and reactor heat removal, or are otherwise essential in preventing significant release of radioactive material to the environment. {NUREG/CR-6421, Section 1.3}

Commercial off-the-shelf (COTS) software: Software defined by market-driven need, commercially available, and whose fitness for purpose has been demonstrated by a broad spectrum of commercial users. {BS EN 50128:2011, Section 3.1.3}

Commercial-grade dedication: An acceptance process undertaken to provide reasonable assurance that a CGI to be used as a basic component will perform its intended safety function and, in this respect, will be deemed equivalent to an item designed and manufactured under a 10 CFR Part 50, Appendix B, QA program. {NUREG/CR-6421, Section 1.3}

See also the 10 CFR 21.3 definition of dedication below; it is the same as commercial-grade dedication in this context. Commercial-grade dedication, when applied to software or software tools, especially COTS, MOTS, or GOTS software or

software tools, is equivalent to the process of “qualification” as described in RTCA DO-330.

Commercial-grade item (CGI): (1) When applied to nuclear power plants licensed pursuant to 10 CFR Part 50, CGI means a structure, system, or component, or part thereof that affects its safety function, that was not designed and manufactured as a basic component. Commercial-grade items do not include items where the design and manufacturing process require in-process inspections and verifications to ensure that defects or failures to comply are identified and corrected (i.e., one or more critical characteristics of the item cannot be verified). {10 CFR 21.3} (2) A structure, system, or component, or part thereof, that is used in the design of a nuclear power plant and which could affect the safety function of the plant, but was not designed and manufactured as a basic component. {NUREG/CR-6421, Section 1.3}

Note that the current rulemaking process for revision of 10 CFR Part 21 is expected to add nuclear power plants licensed pursuant to 10 CFR Part 52 to those for which this definition of CGI is applicable. The Part 21 revision is also expected to clarify the statement that “Commercial-grade items do not include items where the design and manufacturing process require in-process inspections and verifications to ensure that defects or failures to comply are identified and corrected (i.e., one or more critical characteristics of the item cannot be verified).” The clarification is expected to make it clearer that items requiring in-process inspections and verifications may be dedicated, provided that all critical characteristics can be verified. Such verification would be done through a commercial-grade survey (EPRI Method 2) that verifies that the manufacturer verifies the critical characteristics in question during manufacture (e.g., at hold points) or by means of source verification (EPRI Method 3) in which a representative of the dedicating entity performs or witnesses the verification directly (at hold points) on the actual CGIs being dedicated and supplied by or through the dedicating entity (end user or third-party).

Common-cause failure (CCF): Loss of function to multiple structures, systems, or components due to a shared root cause. This is important to an understanding of software tools as the same tool may be used in development of software for multiple systems thus making a fault in the tool a potential source of a common-cause failure. {IEEE 603-2009, Clause 3}

Complexity: The degree to which a system or component has a design or implementation that is difficult to understand and verify. {IEEE 100-2000}

Component: A constituent part of software which has well defined interfaces and behavior with respect to the software architecture and design, covers a specific subset of software requirements, and is clearly identified and has an independent version inside the configuration management (CM) system or is a part of a collection of components (e.g., subsystems) which have an independent version: {BS EN 50128:2011, Section 3.1.4}

Configuration item: (1) One or more hardware or software components treated as a unit for CM purposes. (2) Software life cycle data treated as a unit for CM purposes. {RTCA DO-178C}

Configuration management (CM): (1) The process of (a) identifying and defining the configuration items of a system, (b) controlling the release and change of these items throughout the software life cycle, (c) recording and reporting the status of configuration items and problem reports, and (d) verifying the completeness and correctness of configuration items. (2) A discipline applying technical and administrative direction and surveillance to (a) identify and record the functional and physical characteristics of a configuration item, (b) control changes to those characteristics, and (c) record and report change control processing and implementation status. {RTCA DO-178C}

Critical characteristics: (1) Identifiable and measurable attributes and variables of a CGI, which once selected to be verified, provide reasonable assurance that the item received is the item specified. {EPRI NP-5652, Glossary} (2) When applied to nuclear power plants licensed pursuant to 10 CFR Part 50, critical characteristics are those important design, material, and performance characteristics of a CGI that, once verified, will provide reasonable assurance that the item will perform its intended safety function. {10 CFR 21.3}

Note that the current rulemaking process for revision of 10 CFR Part 21 is expected to add nuclear power plants licensed pursuant to 10 CFR Part 52 to those for which this definition of critical characteristics is applicable.

Critical characteristics for design: The identifiable and measurable attributes of a replacement item which provides assurance that the replacement item will perform its design function. {EPRI TR-102260, Section 2.2.1.2}

Note that NRC Generic Letter (GL) 91-05 promulgated the NRC staff position that did not distinguish between critical characteristics for design and critical characteristics for acceptance. Rather it referred to the requirements of Criterion III, "Design Control," and Criterion VII, "Control of Purchased Material, Equipment, and Services," that licensees must assure the suitability of all material, equipment, and services for their intended safety-related applications. As discussed in the cited GL, the NRC staff position was and is that not all design characteristics are critical, but if a characteristic is considered critical (i.e., its verification is essential to providing reasonable assurance that the item will perform its intended safety functions when required under all design-basis conditions), then it must be somehow verified, even if indirectly.

Critical characteristics for acceptance: The identifiable and measurable attributes and variables of a CGI, which once selected to be verified, provide reasonable assurance that the item received is the item specified. {EPRI TR-102260, Section 2.2.1.2}

This position was not endorsed by the NRC in GL 89-02, and the NRC staff position on this EPRI definition was clarified in GL 91-05 as discussed above. The definition of critical characteristics in 10 CFR 21.3 as shown above is the one that must be used in connection with commercial-grade dedication of software tools used in the development of safety-related software for nuclear power plants.

Critical software: Software that is part of, or could affect, the safety function of a basic component or a commercial-grade software item that undergoes commercial-grade dedication. {NUREG/CR-6421, Section 1.3}

Dangerous failure: A failure of an element, subsystem, or system that plays a part in implementing a safety function that: {IEC 61508-4, Section 3.6.7}

- (1) prevents a safety function from operating when required (demand-mode) or causes a safety function to fail (continuous mode) such that the equipment under control is put into a hazardous or potentially hazardous state; or
- (2) decreases the probability that the safety function operates correctly when required.

Dedication (See commercial-grade dedication above): When applied to nuclear power plants licensed pursuant to 10 CFR Part 50, dedication is an acceptance process undertaken to provide reasonable assurance that a CGI to be used as a basic component will perform its intended safety function and, in this respect, is deemed equivalent to an item designed and manufactured under a 10 CFR Part 50, Appendix B, QA program. This assurance is achieved by identifying the critical characteristics of the item and verifying their acceptability by inspections, tests, or analyses performed by the purchaser or third-party dedicating entity after delivery, supplemented as necessary by one or more of the following: commercial-grade surveys; product inspections or witness at hold-points at the manufacturer's facility, and analysis of historical records for acceptable performance. In all cases, the dedication process must be conducted in accordance with the applicable provisions of 10 CFR Part 50, Appendix B. The process is considered complete when the item is designated for use as a basic component. {10 CFR 21.3}

Note that the current rulemaking process for revision of 10 CFR Part 21 is expected to add nuclear power plants licensed pursuant to 10 CFR Part 52 to those for which this definition of dedication is applicable.

Defect: (1) An imperfection or deficiency in a project component where that component does not meet its requirements or specifications and needs to be either repaired or replaced. {IEEE 24765-2010} (2) In the context of equipment, systems, or components (including software) important to safety in NRC-licensed facilities (nuclear power plants in particular), the term "defect" means: {10 CFR 21.3}

- (1) A deviation in a basic component delivered to a purchaser for use in a facility or an activity subject to the regulations in this part if, on the basis of an evaluation, the deviation could create a substantial safety hazard;
- (2) The installation, use, or operation of a basic component containing a defect as defined in this section;
- (3) A deviation in a portion of a facility subject to the early site permit, standard design certification, standard design approval, construction permit, combined license or manufacturing licensing requirements of part 50 or part 52 of this chapter, provided

- the deviation could, on the basis of an evaluation, create a substantial safety hazard and the portion of the facility containing the deviation has been offered to the purchaser for acceptance;
- (4) A condition or circumstance involving a basic component that could contribute to the exceeding of a safety limit, as defined in the technical specifications of a license for operation issued under part 50 or part 52 of this chapter; or
 - (5) An error, omission or other circumstance in a design certification, or standard design approval that, on the basis of an evaluation, could create a substantial safety hazard.

Dependability: A broad concept incorporating various characteristics of digital equipment including reliability, safety, availability, maintainability, and others. {EPRI TR-106439, Section 2}

Development: The process by which user needs are translated into a software product. The process involves translating user needs into software requirements, transforming the software requirements into design and implementing the code. Sometimes includes the verification, validation, and support processes. {AECL CE-1001-STD, Section 6}

Deviation: A departure from the technical requirements included in a procurement document, or specified in early site permit information, a standard design certification or standard design approval. {10 CFR 21.3}

Element: Part of a subsystem comprising a single component or any group of components that performs one or more element safety functions. An element may comprise hardware or software (e.g., sensor, programmable controller), or both. {IEC 61508-4, Section 3.4.5}

Equipment under control (EUC): Equipment, machinery, apparatus, or plant used for manufacturing, process, transportation, medical, or other activities. {IEC 61508-4, Section 3.2.1}

Error: (1) With respect to software, a mistake in requirements, design, or code. {RTCA DO-178C} (2) An incorrect state of hardware, software, or data resulting from a fault. {NUREG/CR-6101} (3) The difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. {IEEE 100-2000} (4) A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. {IEC 61508-4, Section 3.6.11} (5) A defect, mistake, or inaccuracy that could result in failure or deviation from the intended performance or behavior. {BS EN 50128:2011, Section 3.1.9}

EUC control system: System that responds to input signals from the process or from an operator and generates output signals causing the EUC to operate in the desired manner. {IEC 61508-4, Section 3.3.3}

EUC risk: Risk arising from the EUC or its interaction with the EUC control system. {IEC 61508-4, Section 3.1.9}

Failure: (1) The inability of a system or system component to perform a required function within specified limits. A failure may be produced when a fault is encountered. {RTCA DO-178C} (2) The external manifestation of an error as seen by a (human or physical device) user, or by another program. {NUREG/CR-6101} (3) The termination of the ability of a functional unit to provide a required function or operation of a functional unit in a way other than as required. {IEC 61508-4, Section 3.6.4} (4) Unacceptable difference between required and observed performance. {BS EN 50128:2011, Section 3.1.10}

Fault: (1) A manifestation of an error in software. A fault, if it occurs, may cause a failure. {RTCA DO-178C} (2) A deviation of the behavior of a computer system from the authoritative specification of its behavior. {NUREG/CR-6101} (3) An abnormal condition that may cause a reduction in, or loss of, the capability of a functional unit to perform a required function. Failures are either random (hardware) or systematic (hardware or software). {IEC 61508-4, Section 3.6.1} (4) A defect, mistake, or inaccuracy that could result in failure or deviation from the intended performance or behavior. {BS EN 50128:2011, Section 3.1.9}

Fault tolerance: The built-in capability of a system to provide continued correct provision of service as specified, in the presence of a limited number of hardware or software faults. {BS EN 50128:2011, Section 3.1.11}

Field-programmable gate array (FPGA): A device containing many circuits whose interconnections and functions are programmable by the user. {IEEE 100-2000}

Functional safety: Part of the overall safety relating to the EUC and the EUC control system that depends on the correct functioning of the safety-related systems and other risk reduction measures. {IEC 61508-4, Section 3.1.12}

Functional safety assessment: An investigation, based on evidence, to judge the functional safety achieved by one or more safety-related systems and other risk reduction measures. {IEC 61508-4, Section 3.8.3}

Hardware safety integrity: Part of the safety integrity of a safety-related system relating to random hardware failures in a dangerous mode of failure (i.e., those failures of a safety-related system that would impair its safety integrity). {IEC 61508-4, Section 3.5.7}

Harm: Physical injury or damage to the health of people or damage to property or the environment. {IEC 61508-4, Section 3.1.1}

Hazard: Potential source of harm {IEC 61508-4, Section 3.1.2}

Hazardous event: Event that may result in harm. Whether or not a hazardous event results in harm depends on whether people, property, or the environment are exposed to the consequence of the hazardous event and, in the case of harm to people, whether any such exposed people can escape the consequences of the event after it has occurred. {IEC 61508-4, Section 3.1.4}

Impact analysis: The activity of determining the effect that a change to a function or component in a system will have to other functions or components in that system as well as to other systems. {IEC 61508-4, Section 3.7.5}

Important to safety: (1) Comprising equipment that (b)(1) is safety-related, (b)(2) is not safety-related, but the failure of which could impact safety, and (b)(3) certain post-accident monitoring equipment as described in Regulatory Guide 1.97. {10 CFR 50.49(b)} (2) A structure, system, or component (a) whose failure could lead to a significant radiation hazard, (b) that prevents anticipated operational occurrences from leading to accident conditions, or (c) that is provided to mitigate consequences of failure of other structures, systems, or components. {NUREG/CR-6421, Section 1.3; IEC 61226, Section 3.10; and IAEA Safety Glossary:2007}

Mechatronics: The synergistic combination of precision mechanical engineering, electronic control, and systems thinking in the design of products and manufacturing processes. {Elsevier Journal}

Mistake: A human action that produces an unintended result or an incorrect result. {IEEE 100-2000}

Mode of operation: The way in which a safety function operates, which may be either: {IEC 61508-4, Section 3.5.14}

Low-demand mode: where the safety function is only performed on demand in order to transfer the EUC into a specified safe state, and where the frequency of demands is **no** greater than one per year; or

High-demand mode: where the safety function is only performed on demand in order to transfer the EUC into a specified safe state, and where the frequency of demands is greater than one per year; or

Continuous mode: where the safety function retains the EUC in a safe state as part of normal operation.

Non-hazardous stable state: A state of a nuclear power plant where stabilization of any transient has been achieved, the reactor is subcritical, adequate heat removal is ensured, and radioactive releases are limited. {IEC 61226, Section 3.11}

Open-source software: Source code available to the general public with relaxed or non-existent copyright restrictions. {BS EN 50128:2011, Section 3.1.18}

Other risk reduction measure: A measure to reduce or mitigate risk that is separate and distinct from, and does not use, safety-related systems. {IEC 61508-4, Section 3.4.2}

Partitioning: A technique for providing isolation between software components to contain and isolate faults. {RTCA DO-178C}

Pre-existing software: Software developed prior to the application currently in question, including COTS and open source software. {BS EN 50128:2011, Section 3.1.17}

Probability of dangerous failure on demand: Safety unavailability of a safety-related system to perform the specified safety function when a demand occurs from the EUC or EUC control system. {IEC 61508-4, Section 3.6.17}

Process: (1) A collection of activities performed in the software life cycle to produce a definable output or product. {RTCA DO-178C} (2) A set of interrelated activities, which transforms inputs into outputs. {IEEE 100-2000}

Programmable electronic: Based on computer technology which may comprise hardware, software, and of input and output units covering microelectronic devices based on one or more central processing units together with associated memories, etc. (e.g., microprocessors, micro-controllers, programmable controllers, ASICs, programmable logic controllers, other computer-based devices like smart sensors, transmitters, and actuators). {IEC 61508-4, Section 3.2.12}

Qualification: (1) As used in the aerospace industry, qualification is a systematic process through which software tools are verified to be suitable for their intended use(s) and to the extent possible, to be free of deficiencies that could adversely affect critical DI&C system software or result in failure to detect deficiencies in that software. (2) In the commercial nuclear industry, in accordance with 10 CFR Part 21, the corresponding process is called *commercial-grade dedication* or simply *dedication* (the same thing in this context). Software tools of the type of concern, intended for use with safety-related software for nuclear power plants, but not designed or developed under a 10 CFR Part 50, Appendix B-compliant QA program (including COTS, MOTS, and GOTS software tools) must successfully complete the dedication process.

Quality objective: A general statement about the relative importance of the features and characteristics of a product of the development process that determine its ability to satisfy requirements. {AECL CE-1001-STD, Section 6}

Random hardware failure: A failure, occurring at a random time, which results from one or more of the possible degradation mechanisms in the hardware. {IEC 61508-4, Section 3.6.5}

Reasonable assurance: A justifiable level of confidence based on objective and measurable facts, actions, or observations from which adequacy may be inferred. {EPRI TR-102260, Section 5}

Reliability: (1) The degree to which a software system or component operates without failure considering only the existence of failure and not the consequences of failure. {NUREG-0800, BTP 7-14, Section A.3.2.1} (2) The ability of a system, component, or item to perform its required functions under stated conditions for a specified period of time. {IEEE 100-2000} (3) A quality objective that requires that the system perform its required behavior such that the probability of it successfully performing that behavior is consistent with the reliability requirements identified. {AECL CE-1001-STD, Section 6} (4) The ability of an item to perform a required function under given conditions for a given period of time. {BS EN 50128:2011, Section 3.1.22}

Reliability hypothesis: The collection of assumptions used to develop the statistical (e.g., first order Markov) model which allows an upper limit failure probability with associated confidence

level to be assigned to a software system based on a number of tests. The major assumptions include: (1) there are only successful test trials, (2) all tests are independent, (3) each test has an equal failure probability, (4) all failures are detected during testing, and (5) the test usage profile is an accurate representation of the expected in-situ demand usage profile. {AECL CE-1001-STD, Section 6}

Reliability qualification: A testing methodology in which a reliability hypothesis is demonstrated to the degree of confidence required, using randomized input values. {AECL CE-1001-STD, Section 6}

Reliability requirements: A requirement specifying the probability of not encountering a sequence of inputs which lead to a failure. {AECL CE-1001-STD, Section 6}

Requirement: (1)(a) A characteristic that a system or a software item is required to possess in order to be acceptable to the acquirer. (b) A binding statement in a standard or another portion of the contract. (2) A statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable, or measurable, and necessary for product or process acceptability. (3)(a) A condition or capability needed by a user to solve a problem or achieve an objective. (b) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document. (c) A documented representation of a condition or capability as in definition (3)(a) or (3)(b). {IEEE 100-2000}

A requirement in this sense is not to be construed as a regulatory requirement (i.e., a provision of NRC regulations). Requirements in this sense, such as mandatory clauses in a standard that must be complied with in order for the standard to be considered met, would normally be characterized as acceptance criteria in NRC review standards. However, compliance with an NRC regulatory guidance document (typically a regulatory guide (RG)) that endorses a standard may be required by one or more regulations, in which case, a requirement of that standard may indirectly become a regulatory requirement.

Reusable software component: Software, its supporting software life cycle data, and other supporting documentation being considered for reuse. The component designated for reuse may be any collection of software, such as libraries, operating systems, or specific system software functions. {FAA AC 20-148, Appendix 1}

Reverse engineering: The process of developing higher-level software data from existing software data. Examples include developing source code from object code or executable object code, or developing high-level requirements from low-level requirements. {RTCA DO-178C}

Risk: (1) Combination of the probability of occurrence of harm and the severity of that harm. {IEC 61508-4, Section 3.1.6} (2) Combination of the rate of occurrence of accidents and incidents resulting in harm (caused by a hazard) and the degree of severity of that harm. {BS EN 50128:2011, Section 3.1.26}

Robustness: The ability of an item to detect and handle abnormal situations. {BS EN 50128:2011, Section 3.1.23}

Safety: Freedom from unacceptable risk. {IEC 61508-4, Section 3.1.11}

Safety critical: A term describing any condition, event, operation, process, equipment, or system that could cause or lead to severe injury, major damage, or mission failure if performed or built improperly, or allowed to remain uncorrected. {NASA-STD-8719.13C, Section 3.2}

Safety critical software: Category I software that is critical to nuclear safety. Failure of software in this category can result in either a system with a high safety related reliability requirement (such as a special safety system) not meeting its minimum performance requirements, or a serious initiating event in a process system (i.e., an event for which the initiating-event frequency limit is very low). {AECL CE-1001-STD, Section 6}

Safety function: Function to be implemented by a safety-related system or other risk reduction measures that is intended to achieve or maintain a safe state for the EUC, with respect to a specific hazardous event. {IEC 61508-4, Section 3.5.1}

Safety integrity: Probability of a safety-related system satisfactorily performing the specified safety functions under all the stated conditions within a stated period of time. Safety integrity comprises hardware safety integrity and systematic safety integrity. {IEC 61508-4, Section 3.5.4}

Safety integrity level (SIL): Discrete level (one out of a possible four), corresponding to a range of safety integrity values, where SIL 4 has the highest level of safety integrity and SIL 1 has the lowest. An SIL is not a property of a system, subsystem, element, or component. The correct interpretation of the phrase “SIL n safety-related system” (where n is 1, 2, 3, or 4) is that the system is potentially capable of supporting safety functions with an SIL up to n . {IEC 61508-4, Section 3.5.8}

Safety-related: (1) A subset of things important to safety; structures, systems, equipment and components that are relied upon to remain functional during and following design-basis events to ensure (a) the integrity of the reactor coolant pressure boundary, (b) the capability to shut down the reactor and maintain it in a safe shutdown condition, or (c) the capability to prevent or mitigate the consequences of accidents that could result in potential offsite exposures {10 CFR 21.3} (2) Pertaining to systems important to safety but that are not safety systems. {NUREG/CR-6421, Section 1.3; IEC 61226, Section 3.17; and IAEA Safety Glossary:2007}

Note that the NUREG/CR-6421, IEC 61226, and IAEA definitions are inconsistent with the definition of the term “safety-related” in several places in NRC regulations, including 10 CFR 50.2, 10 CFR 50.49, 10 CFR 50.65, and 10 CFR 21.3 (as part of the definition of a basic component). These NRC regulations define things safety-related as a subset of things important to safety, specifically as “safety systems” are defined below with the exception that the offsite exposure-related regulations have been updated to include 10 CFR 50.34(a)(1), 10 CFR 50.67(b)(2), and specifying §100.11 of 10 CFR 100. See the definition of “important to safety” above.

Safety-related software: Software that is used to implement safety functions in a safety-related system. {IEC 61508-4, Section 3.5.13}

Safety-related system: A designated system that both: {IEC 61508-4, Section 3.4.1}

- (1) implements the required safety functions necessary to achieve or maintain a safe state for the equipment under control; and
- (2) is intended to achieve, on its own or with other safety-related systems and other risk reduction measures, the necessary safety integrity for the required safety functions.

Safety significant function: A function whose degradation or loss could result in a significant adverse effect on defense-in-depth, safety margin, or risk. {10 CFR 50.69, Section (a)}

Safety systems: (1) Those systems that are relied upon to remain functional during and following design-basis events to ensure (a) the integrity of the reactor coolant pressure boundary, (b) the capability to shut down the reactor and maintain it in a safe shutdown condition, or (c) the capability to prevent or mitigate the consequences of accidents that could result in potential offsite exposures comparable to the 10 CFR Part 100 guidelines. {NUREG/CR-6421, Section 1.3} (2) A system important to safety, provided to ensure the safe shutdown of the reactor and the residual heat removal from the core, or to limit the consequences of anticipated operational occurrences and design basis accident. {IEC 61226, Section 3.18}

Note that in NRC regulations, in the definition of things that are safety-related, which is the same as the NUREG/CR-6421, IEC 61226, and IAEA definitions of safety systems, the offsite exposure-related regulations have been updated to include 10 CFR 50.34(a)(1), 10 CFR 50.67(b)(2), and specifying §100.11 of 10 CFR 100.

Secure development environment: The condition of having appropriate physical, logical, and programmatic controls during the system development phases (i.e., concepts, requirements, design, implementation, and testing) to ensure that unwanted, unneeded, and undocumented functionality (e.g., superfluous code) is not introduced into digital safety systems. {RG 1.152}

Secure operational environment: The condition of having appropriate physical, logical, and administrative controls within a facility to ensure that the reliable operation of digital safety systems is not degraded by undesirable behavior of connected systems and events initiated by inadvertent access to the system. {RG 1.152}

Soft-error: Erroneous changes to data content, but no changes to the physical circuit itself. {IEC 61508-4, Section 3.6.12}

Software: (1) Computer programs and, possibly, associated documentation and data pertaining to the operation of a computer system. {RTCA DO-178C} (2) Intellectual creation comprising the programs, procedures, data, rules, and any associated documentation pertaining to the operation of a data processing system. {IEC 61508-4, Section 3.2.5}

Software level: The designation that is assigned to a software component as determined by the system safety assessment process. The software level establishes the rigor necessary to demonstrate compliance with requirements. {RTCA DO-178C}

Software level reduction: A process used to reduce the rigor necessary to demonstrate compliance with requirements. A lower software level places less emphasis on verification of source code, low-level requirements and software architecture. A lower software level also places less emphasis on the degree of test coverage, control of verification procedures, independence of verification process activities, overlapping verification process activities, robustness testing, and verification activities with an indirect effect on error prevention or detection. {RTCA DO-178C, Section 6.0}

Software life cycle: (1) An ordered collection of processes determined by an organization to be sufficient and adequate to produce a software product. (2) The period of time that begins with the decision to produce or modify a software product and ends when the product is retired from service. {RTCA DO-178C}

Software off-line support tool: Software tool that supports a phase of the software development life cycle and that cannot directly influence the safety-related system during its run time. Software off-line tools may be divided into the following classes: {IEC 61508-4, Section 3.2.11}

- T1: generates no outputs which can directly or indirectly contribute to the executable code (including data) of the safety-related system (e.g., text editor, requirements or design support tool with no automatic code generation capabilities, configuration control tools);
- T2: supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software (e.g., test harness generator, test coverage measurement tool, and static analysis tool);
- T3: generates outputs which can directly or indirectly contribute to the executable code of the safety-related system (e.g., an optimizing compiler where the relationship between the source code program and the generated object code is not obvious, a compiler that incorporates an executable run-time package into the executable code).

Software on-line support tool: Software tool that can directly influence the safety-related system during its run time. {IEC 61508-4, Section 3.2.10}

Software safety integrity: Part of the safety integrity of a safety-related system relating to systematic failures in a dangerous mode of failure that are attributable to software. {IEC 61508-4, Section 3.5.5}

Software safety integrity level: (1) Systematic capability of a software element that forms part of a subsystem of a safety-related system. {IEC 61508-4, Section 3.5.10} (2) Classification

number which determines the techniques and measures that have to be applied to software. {BS EN 50128:2011, Section 3.1.37}

Software tools: A computer program used in the design, development, testing, review, analysis, or maintenance of a program or its documentation. Examples include compilers, assemblers, linkers, comparators, cross-reference generators, decompilers, editors, flow charters, monitors, test case generators, integrated development environments, and timing analyzers. {IEEE 7-4.3.2-2010, Clause 3.1.23}

Special safety system: These are the shutdown systems, the emergency coolant injection system, and the containment system of a CANDU® nuclear generating station. {AECL CE-1001-STD, Section 6}

State of the practice: The most advanced documented knowledge in an engineering field in actual use on a regular basis.

- (1) A study of the “state-of-the-practice” in a particular field may entail a combination of review of literature such as public consensus standards, survey of leading practitioners, interviews with the best responders, and analysis and organization of the acquired data.
- (2) “State-of-the-practice” in an engineering field excludes a process requiring exceptional artistry and craftsmanship (e.g., knowledge not transferable in a written form).
- (3) Distinction between “state-of-the-practice” and “common practice”: The latter is characterized by breadth of usage in a particular engineering field and industry. The former is distinguished through repeated usage of the most advanced and proven knowledge – also known as “best practice” in some communities.
- (4) Distinction between “state of practice” and “state of the art”: The latter term characterizes advanced or recently developed methods, techniques/technologies, approaches, etc., that may still be under development or refinement (e.g., “beta testing”), or that may have been demonstrated to be valid or useful, but have not yet been universally adopted in industry codes or standards, or are not yet in common use in the industry.

Subsystem: Entity of the top-level architectural design of a safety-related system where a dangerous failure of the subsystem results in a dangerous failure of the safety function. {IEC 61508-4, Section 3.4.4}

System: A collection of hardware and software components organized to accomplish a specific function or set of functions. {RTCA DO-178C}

System software: Part of the software of a programmable electronic (i.e., computer-based) system that relates to the functioning of, and services provided by, the programmable device

itself, as opposed to the application software that specifies the functions that perform a task related to the safety of the EUC. {IEC 61508-4, Section 3.2.6}

Systematic capability: A measure (expressed on a scale of SC 1 to SC 4) of the confidence that the systematic safety integrity of an element meets the requirements of the specified SIL, with respect to the specified element safety function, when the element is applied in accordance with the instruction specified in the compliant item safety manual for the element. {IEC 61508-4, Section 3.5.9}

Systematic failure: A failure, related in a deterministic way to a certain cause, which can only be eliminated by a modification of the design or of the manufacturing process, operational procedures, documentation or other relevant factors. {IEC 61508-4, Section 3.6.6}

Systematic safety integrity: Part of the safety integrity of a safety-related system relating to systematic failures in a dangerous mode of failure. Systematic safety integrity cannot usually be quantified (as distinct from hardware safety integrity which usually can). {IEC 61508-4, Section 3.5.6}

System safety integrity level: Classification number which indicates the required degree of confidence that an integrated system comprising hardware and software will meet its specified safety requirements. {BS EN 50128:2011, Section 3.1.39}

Target failure measure: Target probability of dangerous mode failures to be achieved in respect of the safety integrity requirements, specified in terms of either: {IEC 61508-4, Section 3.5.15}

- (1) the average probability of a dangerous failure of the safety function on demand, (for a low-demand mode of operation), or
- (2) the average frequency of a dangerous failure [hr^{-1}] (for a high-demand mode of operation or a continuous mode of operation).

Target risk: Risk that is intended to be reached for a specific hazard taking into account the EUC risk together with the safety-related systems and the other risk reduction measures. {IEC 61508-4, Section 3.1.10}

Task: The basic unit of work from the standpoint of a control program. {RTCA DO-178C}

Tool qualification: A process necessary to obtain certification credit for a software tool within the context of a specific airborne system. {RTCA DO-178C}

Traceability: An association between items, such as between process outputs, between an output and its originating process, or between a requirement and its implementation. {RTCA DO-178C}

User-modifiable software: Software intended for modification without review by the certification authority, the airframe manufacturer, or the equipment vendor, if within the modification constraints established during the original certification project. {RTCA DO-178C}

Validation: (1) The process of determining that the requirements are the correct requirements and that they are complete. The system life cycle processes may use software requirements and derived requirements in system validation. {RTCA DO-178C} (2) Confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled. {IEC 61508-4, Section 3.8.2} (3) Process of analysis followed by a judgment based on evidence to determine whether an item (e.g., process, documentation, software, or application) fits the user needs, in particular with respect to safety and quality and with emphasis on the suitability of its operation in accordance to its purpose in its intended environment. {BS EN 50128:2011, Section 3.1.46}

Verification: (1) The evaluation of the outputs of a process to ensure correctness and consistency with respect to the inputs and standards provided to that process. {RTCA DO-178C} (2) Confirmation by examination and provision of objective evidence that the requirements have been fulfilled. {IEC 61508-4, Section 3.8.1} (3) Process of examination followed by a judgment based on evidence that output items (process, documentation, software, or application) of a specific development phase fulfils the requirements of that phase with respect to completeness, correctness, and consistency. {BS EN 50128:2011, Section 3.1.48}

1 Introduction

Current and future nuclear power plants will rely on digital (computer-based) instrumentation and control (I&C) systems and other types of software-driven systems, which have been and are being developed using various types of software tools. Tool-automated and tool-assisted engineering activities may result in fewer defects than with manual engineering activities if tools are well designed, carefully developed, effectively tested, and appropriately used. However, unknown deficiencies in automated tools or tool-assisted engineering activities may affect the safety assurability of a digital safety-related system, dependent on such tools.

As defined in Institute of Electrical and Electronic Engineers (IEEE) Standard 7-4.3.2-2010, software tools are computer programs used in the design, development, testing, review, analysis, or maintenance of a program or its documentation. Based on industry usage, Information Systems Laboratories, Inc. (ISL) considers that software tools are computer programs or computer-based engineering environments used in the specification of requirements, architecture, design, implementation, testing, and other forms of verification, review, analysis, validation, and documentation of a system or such components of the system as software, and maintenance of these artifacts or work products. Examples include compilers, assemblers, linkers, comparators, cross-reference generators, decompilers, editors, flow charters, monitors, test case generators, integrated development environments, timing analyzers, and code generators, among many others.

The Nuclear Regulatory Commission (NRC) anticipates that more software development applications may include the use of software tools, and the existing guidance may not be adequate for consistent application. NRC Contract NRC-HQ-13-C-04-0004 is a research project intended to develop the technical basis for improved regulatory guidance that will follow industry standards and support NRC regulatory requirements. The improved regulatory guidance should be consistently implementable. The NRC seeks to document existing regulatory guidance, standards, practices, and experience concerning review and approval of the use of software tools in engineering systems in industries across the world. Most of the existing documents are not specifically designed for software tool reviews and the guidance is not always directly relevant to software tools.

1.1 Background

This research project is phased in three tasks. In Task 1, ISL, the contractor, surveyed standards, regulatory guidance, review and approval practices, and other current industry practices concerning the use of software tools on which the safety assurability of a digital, safety-related system is dependent. The product of Task 1 was a report summarizing the results of the survey including a broad baseline of requirements covered by all the industry standards and a broad baseline of guidance, review and approval practices covered by all the regulatory guidance and industry review and approval practices. In Task 2, ISL has analyzed the Task 1 findings and baselines, and modified the baselines for NRC use. The product of Task 2 is this report on the analysis of the findings of Task 1. It includes an assessment of the hazards associated with the use of each type of software tool in different stages of the

engineering life cycle (e.g., challenges to assurability, verifiability, and analyzability) and corresponding criteria to qualify the tools for representative uses. In Task 3, ISL will establish a technical basis for influencing public consensus standards and to incorporate review criteria in NRC regulatory guidance. The product of Task 3 will be a corresponding report.

As part of Task 1, ISL researched industry standards, guidance, and practices to determine the types of software tools used in the nuclear and other industries and the general philosophy for software tool use during software development. ISL prepared a generic list of software tools and created a list of software tool categories based upon tools that are widely used within the software development industry. Based on similarities between certain software tools and software tool usage and availability in certain life cycle processes, all software tools have been grouped into nine software tool categories as documented in the Task 1 report.

The Task 1 survey found little information pertaining to software tools in the nuclear field internationally. While some guidance was found in other industries, Task 1 concluded that software tools were still not widely addressed or regulated except for the civil aviation industry, which contains several documents providing guidance and review and approval practice for software tool use, and IEEE standards that discuss the abundance of computer-aided software engineering (CASE) tools. Based on the extensive survey of major industries, ISL concluded in Task 1 that the IEEE standards related to the selection, use, evaluation, and qualification of CASE tools along with the Federal Aviation Administration (FAA) standards, FAA orders, FAA advisory circulars, Certification Authorities Software Team (CAST) papers, and Radio Technical Commission for Aeronautics (RTCA) documents, should provide a basis upon which to build software tool (including commercial off-the-shelf (COTS), government off-the-shelf (GOTS), and modified off-the-shelf (MOTS) software tools) regulation, guidance, and review and approval practices for the commercial nuclear industry pending further analysis.

Based on the information in all of the documents surveyed in Task 1, ISL found that software tools were used in many of the software and system life cycle processes; however, guidance and review and approval practices were typically limited to software tools used in processes, activities, and tasks that have the potential of introducing defects or failing to detect defects in the final safety-related software. ISL considered that software tool regulatory guidance and review and approval practices should be expanded to cover more software life cycle processes when assurability is dependent on that tool. Based on the activities, tasks, and products associated with the system-context and software-specific processes of IEEE 12207-2008, and whether a tool used during a specific process impacts the final nuclear plant software, ISL was persuaded that some of the system-context processes and most of the software-specific processes should be subject to regulatory guidance.

In Task 1, ISL produced a broad baseline of requirements covering all the industry standards for the use of software tools (including COTS, GOTS, and MOTS software tools) and a broad baseline of guidance and review and approval practice covering all the industry guidance. The baseline requirements for software tools to support regulatory assurance engender the need for proper evaluation and qualification, adequate configuration management (CM), thorough verification and validation (V&V), proper use as described by the tool's intended purpose,

correct software tool classification level, thorough up-front planning, continuous quality assurance (QA), adequate reviews, and adequate training on tool use. The baseline guidance for software tools to support regulatory assurance include evaluation and qualification guidance, CM practices, V&V practices, guidance and limitations on tool use, tool classification level guidance, planning practices, QA guidance, review and approval guidance, and training practices.

1.2 Overview

This report is the product of contract NRC-HQ-13-C-04-0004 Task 2. ISL analyzed and compared numerous documents from the industries surveyed as part of Task 1 including the aerospace industry, civil aviation industry, and national commercial nuclear power industry. ISL also expanded the analysis to the automotive, railway, and international commercial nuclear power industries. ISL also analyzed and compared numerous documents from several organizations including the NRC, IEEE, International Electrotechnical Commission (IEC), International Atomic Energy Agency (IAEA), Digital Instrumentation and Control Working Group (DICWG) of the Multinational Design Evaluation Program (MDEP) of the Nuclear Energy Agency (NEA) of the Organization for Economic Cooperation and Development (OECD), Electric Power Research Institute (EPRI), National Institute of Standards and Technology (NIST), International Organization for Standardization (ISO), European Committee for Electrotechnical Standardization, and Atomic Energy of Canada, Ltd (AECL). The analysis focused on standards, regulatory guidance, and review and approval practices related to the use of software tools developed in-house, software tools developed by a contractor, COTS software tools, GOTS software tools, and MOTS software tools. This report summarizes the analysis of each industry or organization with specific emphasis on discussions of software tools.

Intentionally Blank

2 Document Analysis

This section contains the results of the analysis of existing standards, guidance, and review and approval practices associated with various industries and organizations. In the context of this document, analysis means the separation of all the documents in a given industry or organization into its constituent elements, determining the essential features of each document, and how, collectively, the documents relate to one other to address software tools. Section 2 is subdivided into 11 subsections to discuss different organizations and industries. Each subsection discusses the analysis of standards, guidance, and review and approval practices of that industry or organization.

2.1 Aerospace Industry Analysis

The National Aeronautics and Space Administration (NASA) conducts work in four principal activities: (1) aeronautics, (2) human exploration and operations, (3) science, and (4) space technology. Specifications and standards are an essential part of these engineering activities and provide NASA engineers with the critical standards and documents needed to perform their work. Technical standards provide the means to achieve and maintain desired levels of compatibility, interchangeability, or commonality within NASA and with industry practices. NASA issues documents for use by NASA headquarters and all NASA centers to provide a common framework for consistent practices across NASA programs. Documents issued by NASA include high-level policy directives (NPD), procedural requirements (NPR), standards, guidebooks, and handbooks.

2.1.1 Document Analysis

There are no NPDs that address software tools. However, six lower-level NASA documents discuss software tools. One of these six documents is NPR 7150.2A. Three of the six documents are NASA standards (i.e., NASA-STD-8739.8, NASA-STD-7009, and NASA-STD-8719.13C). The final two documents include a NASA guidebook (GB), NASA-GB-8719.13, and a NASA handbook, NASA-HDBK-8739.23. The following paragraphs discuss the analysis of these six documents relative to the use, verification, validation, qualification, certification, accreditation, review, and approval of software tools.

2.1.1.1 NASA Procedural Requirement 7150.2A

NPR 7150.2A is the highest-level NASA document relevant to software tools. NPR 7150.2A provides a set of generic software engineering requirements and establishes the NASA software classifications definitions. NPR 7150.2A contains a few software tool requirements; however, most of the software tool discussions simply identify and encourage the use of certain types of tools in different software development life cycle processes. For example, NPR 7150.2A encourages the use of tools, such as rigorous specification languages and graphical representations, in the software design process to prevent and eliminate errors as early in the life cycle as possible. The NPR 7150.2A software engineering requirements relevant to software tools are discussed in the following paragraphs. A cross-reference to NPR 7150.2A

requirements is provided by supplying the requirement number that contains the letters SWE followed by a three digit number.

The first software engineering requirement related to software tools, SWE-020, states that all projects shall classify each system and subsystem containing software or developed using software in accordance with the NPR 7150.2A, Appendix E, software classification definitions. Each system and subsystem containing or developed with software is assigned to a specific software class based on the software usage, criticality of the system, extent to which humans depend upon the system, and extent of NASA investment in the software. Classes A through E cover engineering-related software and Classes F through H cover business and information technology (IT) software. The number of applicable requirements and their associated rigor are scaled back for lower software classes and software designated as nonsafety-critical. The software classification definitions from NPR 7150.2A, Appendix E, are summarized as follows:

- Class A, human-rated software systems, includes all space flight software subsystems that support human activity in space and that interact with human space flight systems. Examples of Class A software include guidance, navigation and control; life support; crew escape; automated rendezvous and docking; failure detection; isolation and recovery; and mission operations. Class A does not include software or systems used to test Class A systems containing software in a development environment (e.g., simulators, emulators, stimulators, and facilities).
- Class B, non-human space-rated software, includes flight and ground software that must perform reliably to accomplish primary mission objectives. Examples include propulsion systems; power systems; guidance, navigation and control; fault protection; thermal systems; command and control ground systems; planetary surface operations; hazard prevention; primary instruments; or other subsystems that could cause the loss of scientific data returning from multiple instruments. Class B does not include software or systems used to test Class B systems containing software in a development environment (e.g., simulators, emulators, stimulators, and facilities).
- Class C, mission support software, includes flight or ground software that is necessary for returning scientific data from a non-critical instrument, for analyzing or processing mission data, or other software for which a defect could adversely impact secondary mission objectives or cause operational problems for which potential workarounds exist. Examples include prelaunch integration and test, mission data processing and analysis, analysis software used in trend analysis and calibration of flight engineering parameters, and primary/major science data collection and distribution systems. Class C does not include systems that are not part of and do not impact a facility or vehicle.
- Class D, analysis and distribution software, includes non-space flight software developed to perform science data collection, storage, and distribution; or

perform engineering and hardware data analysis. A defect in Class D software may cause rework but has no direct impact on mission objectives or system safety. Examples of Class D software include ground engineering design and modeling tools; project assurance databases, analysis and test tools, and science data and distribution systems. Class D does not include software that can impact primary or secondary mission objectives, software that operates a facility or vehicle, or space flight software.

- Class E, development support software, includes non-space software developed to explore a design concept; or support software or hardware development functions such as requirements management, design, test and integration, configuration management (CM), documentation, or perform science analysis. A defect in Class E software may cause rework but has no direct impact on mission objectives or system safety. Class E does not include space flight systems, software supporting an operational system, software directly affecting primary or secondary mission objectives, software that can adversely affect the integrity of scientific artifacts, software used in technical decisions in operational systems, or software that impacts operational vehicles.
- Class F, general purpose computing software, includes software used in support of voice, wide area network, local area network, video, data centers, application services, messaging and collaboration, and public web. A defect in Class F software is likely to affect productivity of multiple users and may possibly affect mission objectives or system safety including cost, schedule, or technical objectives. Examples of Class F software include the NASA web portal and software supporting NASA's financial program such as time and attendance system, travel management, business warehouse and payroll.
- Class G, general purpose computing software, includes software used in support of voice, local-area network, video, data centers, application services, messaging and collaboration, and public web. A defect in Class G software is likely to affect productivity of a single user or small group of users but generally will not affect mission objectives or system safety. Examples of Class G software include the NASA Headquarters' corrective action tracking system and NASA Headquarters' new user request system.
- Class H, general purpose desktop software, includes software for Wintel, Mac, and UNIX desktops as well as laptops. A defect in Class H software may affect the productivity of a single user or small group of users, but generally will not affect mission objectives or system safety. However, a defect in desktop IT-security-related software (e.g., antivirus) may lead to loss of functionality and productivity across multiple users and systems. Examples of Class H software include typical desktop applications such as Microsoft Word, Excel, and PowerPoint and Adobe Acrobat.

An ISL observation, based on our analysis of this first requirement, is that the requirement to classify each system and subsystem containing software, as written, does not apply directly to software tools since the tool itself is not software contained in a system or subsystem of the project. However, in an attempt to clarify the software classification requirement, NPR 7150.2A states that software written to support development activities (e.g., verification tools and simulations) should be classified separately from the system under development. ISL found that the clarification expands the requirement to cover all types of software tool classifications regardless of their origin (e.g., in-house, vendor, university, and contractor). Based on our analysis of the above software classification descriptions, ISL observes that NASA considers analysis software tools to be Class C or D software depending on whether the analysis supports mission objectives. Software development tools are Class E software and are not assigned the same classification as the software being developed.

The second requirement, SWE-132, also relates to software classification and states that the software quality assurance (QA) organization shall perform an independent classification assessment to compare to the classification assigned by the project. Through reasoning consistent with the extension of requirement SWE-020 to software tools, an analysis of requirement SWE-132 suggests that the requirement could also be extended to require an independent classification of software tools. However, while independence is desirable for classifying safety-related software, an ISL observation, based on our analysis of this second requirement, is that the practice is unnecessary because tools typically have a limited functionality and are relatively easy to classify correctly; whereas application software is typically complex and multifunctional.

The next two requirements are related to software safety criticality. SWE-133 states that all projects shall determine the software safety criticality in accordance with NASA-STD-8739.8. NASA-STD-8739.8 contains a software safety litmus test which classifies software as either safety-critical or nonsafety-critical. The designation of the software as safety-critical or nonsafety-critical along with the software classification from NPR 7150.2A, Appendix E, determine which requirements of NPR 7150.2A are applicable to specific systems and subsystems containing software. The second safety requirement, SWE-023, states that when a project is determined to have safety-critical software, the project shall ensure that the safety requirements of NASA-STD-8719.13 are implemented by the project. NPR 7150.2A also states that further scoping and tailoring of the safety effort can follow the NASA guidebook NASA-GB-8719.13. An ISL observation, based on our analysis of these two requirements, is that NASA requires that all projects determine whether software tools are safety-critical or nonsafety-critical in accordance with NASA-STD-8739.8. Based on that determination and the software classification, the requirements of NPR 7150.2A applicable to software tools can be determined.

An overall ISL observation, based on our analysis of these first four requirements, is that the requirements are primarily concerned with and apply to software that resides within a system or subsystem and which will ultimately be executed on the vehicle, space asset, or ground as part of the mission. The requirements are not written specifically with software tools in mind, and the applicability of the requirements to software tools requires a liberal interpretation of words that clarify and expand the requirements. These first four requirements are difficult to apply to

software tools in their current form, and it would be difficult to implement these requirements as a technical basis for regulatory guidance. However, the essential elements of these requirements that could be considered relevant to software tools are that software tools should be classified by the project and an independent QA organization based on their importance to the project and software tools should be classified relative to their safety implications to the project. Based on these two classifications, the amount of rigor necessary to verify, validate, certify, review, and approve the software tool should be determined. Note that while NPR 7150.2A calls for independent classification for tools, ISL, for the reasons stated above, believes that if the tool is rigorously scrutinized by the user, independent classification is unnecessary.

Another software engineering requirement related to software tools, SWE-135, states that during the software implementation process, the software engineering project shall ensure that results from static analysis tool(s) are used in verifying and validating software code. A note associated with this requirement acknowledges the shortcoming of static analysis tools, which can generate false positives when verifying adherence to coding methods, standards, and criteria. However, NPR 7150.2A states that users of the static analysis tool can calibrate, tune, and filter results to make effective use of these tools and the results from static code analysis tools can be reviewed during the review and inspection process to overcome the false positives. An ISL observation, based on our analysis of SWE-135, is that the requirement simply acknowledges the existence of static analysis tools, encourages their use, but doesn't provide any guidance for using, verifying, validating, or approving static analysis tools as part of a software development project. Therefore, ISL recommends that the NRC consider developing regulatory guidance for using, verifying, validating, and approving static analysis tools.

SWE-136 states that all software engineering projects shall validate and accredit software tool(s) required to develop or maintain software. This requirement implies that all software tools must be validated and accredited. However, a note following the requirement states that the requirement only applies to tools used to develop or maintain Class A, B, C, or safety-critical software (i.e., most important to vehicle, crew, or primary and secondary mission functions or objectives). The note following the requirement also states that the validation and accreditation can be accomplished by determining and defining acceptance processes for software tools but does not discuss those acceptance processes. The wording in the note following the requirement implies that if a tool is used to develop or maintain software with a classification of D, E, F, G, or H, and the tool is not used to develop or maintain safety-critical software, then the tool is not required to be validated and accredited. An ISL observation, based on our analysis of SWE-136, is that although the requirement states that tools shall be validated and accredited under certain situations, neither the requirement nor the note following the requirement references any particular document or method for validating and accrediting these tools other than stating that an acceptance process should be defined. The essential element of this requirement is that tools should be validated and accredited when used to develop or maintain Class A, B, C, or safety-critical software.

SWE-066 states that all projects shall perform software testing as defined in the software test plan. Although this requirement does not specifically address tools, a note following the requirement encourages the use of automated software testing tools in the software testing life

cycle process. An ISL observation, based on our analysis of SWE-066, is that the requirement does not apply to assessment of tools, but simply encourages the use of automated testing tools, and NPR 7150.2A does not provide any guidance on the use, verification, validation, review, or approval of automated testing tools. Therefore, ISL recommends that the NRC consider developing or updating regulatory guidance on the use, verification, validation, review, and approval of automated testing tools.

SWE-070 states that all software engineering projects shall verify, validate, and accredit software models, simulations, and analysis tools required to perform qualification of flight software or flight equipment. A note following the requirement states that information regarding specific verification and validation (V&V) techniques and the analysis of models and simulations can be found in NASA-STD-7009. An ISL observation, based on our analysis of SWE-070, is that the requirement applies to software tools beyond those used in the software development process which are covered by SWE-136. These analysis tools support the software design but do not directly generate code or verify, validate, or qualify software. This requirement would apply to tools such as RELAP or TRACE when used to model, simulate, or analyze commercial nuclear plant normal and abnormal transients. Therefore, ISL recommends that the NRC consider developing or updating regulatory guidance for models, simulations, and analysis tools.

The remaining software tool-relevant material in NPR 7150.2A deals with proper configuration control of tools, maintaining a library of best tools, and the content of various plans which need to discuss tools. Specifically, SWE-102 states that the software development or management plan shall contain a discussion of the engineering environment including tools. Also, SWE-103 states that the software configuration management (SCM) plan shall contain a discussion of resource information which identifies the software tools necessary for implementation of the activities of CM. Finally, SWE-105 states that the software maintenance plan shall include a discussion of specific tools associated with the maintenance process. An ISL observation, based on our analysis of these requirements and other industry practices, is that proper configuration control and documentation of software tools is standard industry practice and that current regulatory guidance adequately addressed this practice.

2.1.1.2 *NASA-STD-7009*

NASA-STD-7009 provides a very comprehensive set of requirements for the development, documentation, and operation of models and simulations. However, an ISL observation, based on our analysis of these requirements, is that the requirements are very specific to this one type of software engineering tool and would not fit readily into the nuclear regulatory framework. A possible approach would be for a regulatory guide to adopt some of the requirements of this standard for verifying, validating, and approving software models, simulations, and analysis tools.

2.1.1.3 *NASA-STD-8739.8*

NASA-STD-8739.8 discusses requirements for software assurance and determining whether software is safety-critical. In the following discussion, the requirements of NASA-STD-8739.8

are identified by the requirement paragraph number. The standard contains several requirements directly or indirectly applicable to software tools.

Paragraph 5.1.2.1 states that the software assurance manager shall ensure completion of the software assurance classification assessment which includes a software safety litmus test. The software safety litmus test in Appendix A to NASA-STD-8739.8 determines if the software is safety-critical based on its ability to contribute to a safety hazard. The safety classification of the software (i.e., safety-critical or nonsafety-critical) and the software classification from NPR 7150.2A (i.e., Classes A through H), determine the software assurance level of effort. Safety-critical software requires a full software assurance level of effort. Nonsafety-critical software requires various levels of software assurance depending on the NPR 7150.2A software classification. Class A requires a full software assurance level of effort, and the level of effort decreases to minimal for Class D. The software assurance requirements are not mandatory for Classes E, F, G, and H. The software assurance level of effort also depends on the potential for mission failure, waste of resource investment, and the potential for impact to equipment, facility, or environment.

The safety litmus test used to determine whether software is safety-critical or nonsafety-critical does not seem to apply to tools; therefore, our analysis suggests that if this is not an oversight, NASA has determined that tools cannot be safety-critical. An ISL observation, based on our analysis of NASA-STD-8739.8, and its applicability to nuclear safety-related software, is that the safety-litmus test should be modified and applied to software tools to better determine the level of software assurance required for the tool. Software tools could be determined to be safety-critical if flaws or errors in or malfunctions of the software generated by the tool, or verified by the tool, could contribute to hazards. Another ISL observation is that if tools are acquired from other companies (e.g., computer-aided software engineering (CASE) tools); enforcement of these software assurance requirements on that company would be problematic. Therefore, the software assurance requirements can only effectively be applied to the provider (i.e., developer) who is using the tool for software development and the acquirer or end user who receives the software product. While the cited requirements in NASA-STD-8739.8 mention tools, they are not specifically written with tools in mind and do not provide a readily adoptable and adaptable technical basis for the development of new regulatory guidance.

Another software assurance requirement of NASA-STD-8739.8 that indirectly applies to software tools, given in paragraph 5.1.2.2, is that the software assurance manager shall ensure that projects with safety-critical software comply with the requirements in paragraph 5.1.2.2 of NASA-STD-8719.13. An ISL observation, based on our analysis of this requirement, is that software tools should also comply with the requirements in NASA-STD-8719.13 if the software created by the tool is found to be safety-critical.

Paragraph 6.1.1 states that the provider of the software product shall plan, document, and implement a software assurance program for software development, operation, and maintenance activities including documentation of software assurance procedures, processes, tools, techniques, and methods to be used. ISL analysis of this requirement suggests that the requirement could be interpreted to mean that a software assurance program should include

documentation for only software assurance tools. However, a more reasonable interpretation of the requirement is that tools used for software development, operation, and maintenance activities should be documented. Implementing a QA program, including documentation of tools, is a typical industry practice and is covered by existing regulatory guidance.

Paragraph 6.8.3 states that software assurance personnel shall be trained in relevant software engineering design tools to stay current with the products they must assure. ISL recommends that the NRC should consider that adequate software tool training, not only of software engineering staff, but of also QA personnel, be prescribed or at least designated as an acceptance criterion in pertinent regulatory guidance.

Appendix C of NASA-STD-8739.8 is a requirements compliance matrix. The matrix lists all the requirements of NASA-STD-8739.8 and identifies the title or position of the person who has responsibility for satisfying the requirement. The matrix also includes a compliance checklist where the reviewer can formally document full, partial, or non-compliance with each requirement. The matrix also allows the reviewer to document comments justifying the compliance rating. In this particular matrix, requirements are listed for all software development processes from conception through retirement and for all the supporting processes of QA, training, verification, and validation. This particular matrix covers requirements for software development with no direct application to software tools. An ISL observation, based on our analysis of the concept of a requirements compliance matrix, is that such a matrix would be worthwhile as a software tool review and approval tool. Once software tool requirements appropriate for the commercial nuclear power industry are finalized, the requirements can be listed in a compliance matrix that can be used by the NRC to formally document compliance to the requirements.

2.1.1.4 *NASA-STD-8719.13C*

NASA-STD-8719.13C is a software safety standard that defines the requirements and describes the activities required to ensure and promote safety processes that are utilized for software that is created, acquired, or maintained by or for NASA. This standard only applies to software that is safety-critical as determined by the software safety litmus test in NASA-STD-8739.8, Appendix A. Although applicability to software tools is not clear in other NASA standards, an ISL observation, based on our analysis of NASA-STD-8719.13C, is that the applicability of this standard extends to software tools used to develop, test, verify, or validate safety-critical software even though the safety litmus test is not written to accommodate a safety-critical assessment of tools.

NASA-STD-8719.13C is a significant upgrade of NASA-STD-8719.13B with respect to software tool safety through the addition of a software safety criticality assessment (SSCA). The software safety requirements relevant to software tools are discussed in the following paragraphs. A cross-reference to NASA-STD-8719.13C requirements is provided by supplying the requirement number which contains the letters SSS following by a three digit number.

Section 1.2 of NASA-STD-8719.13C acknowledges that commercial off-the-shelf (COTS) software tools constitute the majority of the tools NASA uses. The section further states that

Sections 6.5, 6.6, and Appendix F of NASA-STD-8719.13C address safety of COTS software and software tools. Appendix F of NASA-STD-8719.13C implies that COTS tools should be treated separately from COTS software since Section F.2 addresses COTS software safety considerations and Section F.3 addresses software tools. ISL recommends that the NRC consider clarifying, in any new regulatory guidance, the distinction between COTS software which becomes part of the final software product and COTS tools which are used in the software development process.

Section 4 of NASA-STD-8719.13C contains a requirement, SSS-001, that states that the acquiring safety and mission assurance (SMA) organization shall perform the initial SSCA, Part 1, software safety litmus test, to determine whether a project has safety-critical software. Although this requirement is not relevant to software tools, it is worth mentioning because the requirement seems superfluous in view of the NASA-STD-8739.8 requirement for a software safety litmus test prior to determining the applicability of NASA-STD-8719.13C. The requirements of NASA-STD-8719.13C are only applicable if the software safety litmus test performed by NASA-STD-8739.8 determines that the system or subsystem contains safety-critical software.

Section 6.5 of NASA-STD-8719.13C encourages the use of tools to produce consistent analyses and code and to find errors. However, the section acknowledges that tool use can inadvertently introduce software hazards into the software or hardware. This section states that tools are defined, accredited, and configuration managed per NPR 7150.2. However, the analysis of NPR 7150.2A, which is the latest version of NPR 7150.2, found little guidance directly applicable to software tools except for one software engineering requirement in NPR 7150.2A, SWE-070, as discussed above, that states that all software engineering projects shall validate and accredit software tool(s) required to develop or maintain software, but it does not provide a procedure for completing those actions. The only guidance in NPR 7150.2A for validating and accrediting software tools is that those tasks can be accomplished by determining and defining acceptance processes for software tools, but it does not discuss those acceptance processes.

Section 6.5 of NASA-STD-8719.13C also states that software tools bought for use in testing; modeling; simulating test scenarios; configuration managing critical data and files; designing hardware; assessing flight, launch, and landing re-entry algorithms; and testing and creating configuration files for a programmable logic device will be looked at as COTS first and foremost. In addition, any created applications running on any of these tools, will also be looked at as COTS first and foremost. Section 6.5 further states that these tools would then be subjected to the black-box COTS level of software safety assurance and states that the software safety assurance approach involves verifying the inputs and outputs of the tool, checking the configuration and version control of the tool, checking the limits of the tools' abilities, and addressing the potential risks of tools not performing as advertised or as programmed. It is not clear whether the four processes in the approach discussed above are just general guidance or whether they constitute the "black-box" processes. The words "first and foremost" are also not clear and could lead to the inference that tools should be treated as COTS software and subject to the same requirements as COTS software. Treating COTS tools as COTS software would be

inconsistent with the treatment in Appendix F of NASA-STD-8719.13C. In addition, treating COTS tools as COTS software would mean that the COTS column in the NASA-STD-8719.13C, Appendix A, applicability matrix would apply to tools even though it is clear that some of the requirements applicable to COTS software are not applicable to tools.

Section 6.5 of NASA-STD-8719.13C contains a requirement relevant to software tools, SSS-018, that states that the provider shall use the SSCA to determine if the tools planned for use on the program, project, and facility have a potential impact on the safety of the system. In this situation, the provider must determine the safety impact on the system because the provider has obtained the tool from a vendor and the vendor, which has developed generalized CASE tools for use in any software development project, has no prior knowledge that the tool would be used in a safety-critical software development project.

An ISL observation, based on an analysis of NASA-STD-8719.13C, is that a SSCA, as described in Appendix A of NASA-STD-8719.13C, needs to be performed for software tools to determine which of the requirements in this standard apply. Part 1 of the SSCA is to perform a software safety litmus test that appears redundant with the same test in NASA-STD-8739.8 and is ineffective because the software safety litmus test is not written to apply to tools. However, the SSCA contains more than just a software safety litmus test. Part 2 of the SSCA is a software risk assessment which assesses the amount of critical software within a system, the level of autonomy of that software, and the time to criticality, as well as the severity of failure. The software risk assessment determines the safety assurance level of rigor. Based on the safety litmus test and the risk assessment, an applicability matrix provides pre-tailoring of requirements by specifying which NASA-STD-8719.13C requirement applies for different levels of risk. Another ISL observation, based on analysis of the SSCA, is that the amount of critical software, level of autonomy, time to criticality, and severity of failure as assessed in Part 2 of the SSCA are not relevant to software tools. However, a risk assessment of tools is required by SSS-018 even though it appears to be inconsistent with previous statements in Section 6.5 that tools should be assessed using the “black box” level of assurance. The applicability of the NASA-STD-8719.13C, Appendix A, SSCA to software tools is not straightforward, and applying COTS software requirements to tools does not seem appropriate. Therefore, this standard does not provide a sound technical basis for improving NRC regulatory guidance.

2.1.1.5 *NASA-GB-8719.13*

NASA-GB-8719.13 is a NASA guidebook that provides an overview of general software safety and good software engineering practices which contribute to software system safety. The guidebook also provides analyses, methods, and guidance which can be applied during each phase of the software life cycle for meeting the requirements of NASA-STD-8719.13C. NASA-STD-8719.13C addresses the “who, what, when, and why” of software safety analysis while NASA-GB-8719.13 addresses the “how to” aspect of software safety analysis. However, NASA-GB-8719.13 provides guidance on the processes, activities, and tasks necessary to satisfy the requirements of NASA-STD-8719.13B and has not been updated to recognize the SSCA procedure added in NASA-STD-8719.13C. Therefore, the guidebook does not contain “how to” information on performing the SSCA, how the SSCA should be conducted for software tools, or

how to use the COTS column in the NASA-STD-8719.13C, Appendix A, applicability matrix for tools. It also lacks any information clarifying how tools should be treated as COTS, first and foremost, as stated in NASA-STD-8719.13C.

This guidebook does contain a wealth of information on software tools, but the information simply identifies specific tools that are available for use in certain software development processes. It also encourages the use of tools in certain software development processes, provides some limited guidance on tool use, and some limited guidance on tool V&V. Reviewing or approving tools is not covered in this guidebook.

Section 4 of NASA-GB-8719.13 discusses the safety-critical software development process and contains a few recommendations for effective tools. Section 4.2.2.2 of NASA-GB-8719.13 recognizes that a unified modeling language (UML) is a language and methodology for specifying, visualizing, and documenting the development artifacts of an object-oriented system. The guidebook states that tools already incorporate UML and some tools can generate code directly from the UML diagram. However, the guidebook does not discuss how these tools are verified, reviewed, or approved for use nor does it provide any guidance for their use.

Section 4.2.2.3 of NASA-GB-8719.13 states that the formal methods (FM) design methodology consists of a set of techniques and tools based on mathematical modeling and formal logic that are used to specify and verify requirements and designs for computer systems and software. However, the guidebook does not discuss FM tool verification, review, approval, or guidance. The guidebook simply provides a number of references which contain information on FM tools.

Section 4.5.2 of NASA-GB-8719.13 provides guidance for using SCM tools including versioning, check-in and check-out constraints, and good documentation of changes which often is lacking when using SCM tools. However, the guidebook does not discuss SCM tool verification, validation, qualification, review, or approval practices.

Section 4.6 of NASA-GB-8719.13 encourages the use of simulators or an in-circuit emulator (ICE) system for debugging in embedded systems because these tools allow the programmer or tester to find some subtle problems more easily. However, the guidebook does not discuss simulator or ICE system verification, validation, qualification, review, or approval practices.

Section 6 of NASA-GB-8719.13 describes developing and analyzing safety requirements for software, typically expressed as functions with corresponding inputs, processes, and outputs, plus additional requirements on interfaces, limits, ranges, precision, accuracy, and performance. Section 6 of NASA-GB-8719.13 recommends the use of several tools to assist in the software requirements process. Section 6.4.1.1 of NASA-GB-8719.13 discusses the requirements specification and provides a list of requirements management tools. However, the guidebook does not address the verification, validation, qualification, review, or approval of these types of tools. It merely states that a requirements management tool can help manage the changes made to many individual requirements, maintain revision histories, and communicate changes to those affected by them. Section 6.6.3.1 of NASA-GB-8719.13 discusses an automated requirement management (ARM) tool, developed at the NASA Goddard Space Flight Center, which is designed to assess requirements that are specified in natural language. The objective

of the ARM tool is to facilitate a project manager's assessment of the quality but not the correctness of a requirements specification document. NASA-GB-8719.13 does not discuss the verification, validation, qualification, review, or approval of the ARM tool nor does it provide guidance for its use.

Section 6.6.5 of NASA-GB-8719.13 discusses the purpose and functionality of model checking tools. However, the guidebook does not provide guidance on using the tools, nor does it discuss verification, validation, qualification, review, or approval of these tools.

Section 8 of NASA-GB-8719.13 discusses tools used in the implementation phase including the use of COTS tools for analyses of source code and cautions the use of unvalidated COTS tools. Specifically, Section 8 of NASA-GB-8719.13 states that there are some commercial tools available which perform one or more of the required source code analyses in a single package. These tools can be evaluated for their validity in performing these tasks, such as logic analyzers and path analyzers. However, unvalidated COTS tools, in themselves, cannot generally be considered valid methods for formal safety analysis. Section 8 of NASA-GB-8719.13 does not provide guidance on how to validate COTS source code analyzers.

Section 9 of NASA-GB-8719.13 discusses the testing software development life cycle process, provides general guidance on the use of testing tools, and states that modeling, simulation, and analysis tools need to be formally verified. Section 9.4.9 of NASA-GB-8719.13 discusses the use of computer models for system verification and states that the use of computer modeling for design evaluation will continue to expand since the cost savings potential from limiting or eliminating prototype testing will continue to drive industry. Section 9 of NASA-GB-8719.13 further cautions the analyst on the use of these tools because the growing dependence on such computer modeling may lead to the potential misuse of these simulations. There is a tendency to assume the software does the analysis correctly and completely. The analyst must do "sanity checks" of the results, as a bare minimum, to verify that the modeling software is functioning correctly. For safety-critical analyses, the modeling tool should be formally verified or tested. In addition, the analyst needs to have knowledge of the proper application of these "virtual design environments". Therefore, users of software simulation programs that analyze safety-critical hardware should be well trained, experienced, and certified, if the software vendor provides certification. All program input parameters such as loads, restraints, and materials properties should be independently verified to assure proper modeling and analysis. Section 9.5.1 of NASA-GB-8719.13 discusses test coverage analysis and identifies a list of automated tools available to aid in the coverage analysis. The general guidance on verifying modeling software is consistent with the requirements of NASA-STD-7009 and ISL recommends that the NRC should consider the use of NASA-STD-7009 as a technical basis for any new regulatory guidance.

Section 11 of NASA-GB-8719.13 discusses various programming languages, operating systems, tools, and development environments being used to create safety-critical software. Issues with compilers, tools, integrated development environments (IDEs), automatic code generation, and operating systems are considered. Section 11.2 of NASA-GB-8719.13 provides an overview of a typical IDE and identifies the minimum set of tools that a software developer

needs. A developer, as a minimum, needs an editor to create the software source code, a compiler to create object code from the source code, a linker to create an executable application from the object code, and a debugger to find the location of defects in the software. Often, these tools come bundled in an IDE where the developer can shift from editing to compiling to linking to debugging, and back to editing, without leaving the programming environment. In an embedded environment, the IDE can include simulators for the target hardware, the ability to download the software generated into the target hardware, and sophisticated debugging and monitoring capabilities.

Section 11 of NASA-GB-8719.13 discusses features of tools that are desirable for efficient software development and how to select the best tools. The section does not discuss tool verification, validation, or review and approval practices. However, Section 11.3 of NASA-GB-8719.13 discusses CASE tools and states that with respect to automatic code generation tools, the correct translation from design to code must be verified for safety-critical software because CASE tool errors still exist even though the human error is eliminated in the translation. Section 11.3 of NASA-GB-8719.13 states that in an ideal world, the CASE tool would be certified to some standard, and the code generated by it would be accepted without review, in the same way that the object code produced by a compiler is often accepted. However, Section 11.3 of NASA-GB-8719.13 states that based on experience, compilers produce errors in the object code. NASA considers that the use of automatically generated code is in its infancy and when the code is safety-critical, or resides in an unprotected partition with safety-critical code, the automatically generated code should be subjected to the same rigorous inspection, analysis, and test as hand-generated code. However, NASA recognizes that in some environments, the automatically generated code may not be accessible. An ISL observation, based on an analysis of Section 11.3 of NASA-GB-8719.13, is that automatically generated safety-critical code should be subjected to rigorous verification including inspection, analysis, and testing recognizing that inspection is only possible if the automatically generated code is in a form that is readable by and accessible to human inspectors.

Section 11.8.3.2 of NASA-GB-8719.13 states that software which tests safety-critical code must also be considered safety-critical. Thus, ISL recommends that the NRC consider, in any new regulatory guidance, that automatically generated code should be scrutinized with the same rigor as hand-generated code.

Section 12 of NASA-GB-8719.13 discusses issues and concerns with off-the-shelf (OTS), reused, and contracted software. Section 12.1 of NASA-GB-8719.13 covers the pros and cons of OTS and reused software. Section 12.2 of NASA-GB-8719.13 discusses issues relating to having custom software created by a contractor, rather than in-house. Both sections concentrate on actual safety-critical software obtained through these sources and not on software tools that are COTS, government off-the-shelf (GOTS), or reused. The section does not discuss software tools nor does it discuss required verification, validation, or review and approval practices.

Appendix H of NASA-GB-8719.13 contains two checklists for improving the safety of OTS software and additional checklists for good programming practices. NASA-GB-8719.13,

Appendix H, Checklist 2 states that tools for automatic code generation have to be independently validated and OTS tool selection should follow the same process as component selection. Also, the tool's purpose should be considered when selecting OTS tools. The software developer should determine whether the results are easy to verify and whether the results of the tool's use will influence decisions that affect safety. NASA-GB-8719.13, Appendix H, Checklist 5, requires the use of version control tools (CM) and a bug tracking tool or database. NASA-GB-8719.13, Appendix H, Checklist 13 requires that the real-time operating system works with ICE, compilers, assemblers, linkers, and source code debuggers and encourages the use of debugging tools to find defects that are harder to find with source-level debuggers. However, NASA-GB-8719.13, Appendix H, does not discuss details of verification, validation, review, or approval of software tools.

2.1.1.6 *NASA-HDBK-8739.23*

NASA-HDBK-8739.23 focuses on complex electronics and tools used in the design process and assurance activities of those devices. Complex electronics encompasses programmable and designable complex integrated circuits. Programmable logic devices can be programmed by the user and range from simple chips to complex devices capable of being programmed on-the-fly. Some types of programmable devices include a field-programmable gate array (FPGA), complex programmable logic device, application-specific integrated circuit (ASIC), and system-on-chip devices. The handbook states that complex electronics are commonly used within NASA systems and sometimes in safety-critical systems. However, the development of assurance activities for complex electronics is lagging behind the pace of the technology. The handbook provides some general suggestions that, if applied, may increase confidence in the quality of complex electronic devices, but it does not provide any explicit procedures for verifying, validating, qualifying, reviewing, or approving tools used in the design process of complex electronics.

Section 5.8.3 of NASA-HDBK-8739.23 identifies that much of the implementation process for complex electronics is performed by automated tools, and if the tools were previously assessed, the results of the implementation process can be accepted without additional review. However, the handbook does not identify how the tools could be assessed other than checking the tool vendor web site and other sources for known tool defects or operational workarounds. The only guidance for complex electronics tools is from NASA-STD-8719.13C, which states that if these tools are defined as safety-critical, then they should be treated as COTS software and all that can be done is to verify the inputs and outputs, check the configuration and version control, check the limits of the tools' abilities, and address the potential risks of their not performing as advertised.

Although the handbook discusses a significant number of the vendor- and third-party-supplied tools used in the design process of complex electronics, the handbook does not discuss QA, V&V, review, or approval of those vendor-supplied tools. The handbook acknowledges that tool-induced design errors occur and can be difficult to detect. Tools are a vital part of complex electronics design, and the designer often does not know what errors a tool could potentially

produce. Further, complex functionality cannot be completely simulated, nor the resulting chip completely tested.

In summary, NASA-HDBK-8739.23 does not provide any tool related verification, validation, qualification, review, or approval processes and instead instructs the designer to rely on the vendor web site and other unnamed sources to determine tool defects and operational workarounds. On the basis of our analysis of this handbook, including its lack of specific tool guidance, this handbook is not suitable as a technical basis for regulatory guidance.

2.1.2 Summary

NASA procedural requirements, standards, guidebooks, and handbooks contain a few high-level software tool requirements, guidance, and review and approval practices that could form a good technical basis for developing new regulatory guidance for the commercial nuclear power industry.

NASA NPR 7150.2A, Appendix E, defines software classifications that determine the required QA effort associated with that class of software or software tools. NPR 7150.2A requires that software be designated as either safety-critical or nonsafety-critical and classified as Class A through H depending on functionality. Engineering software development and analysis support tools and business and IT software are classified within the same classification system as system and subsystem software. Classes A through E cover engineering related software in decreasing order of applicable NPR 7150.2A requirements. Classes F through H cover business and IT software in decreasing order of applicable NPR 7150.2A requirements. Analysis software supporting the mission is classified as Class C software. Ground engineering design and modeling tools and analysis and test tools that cannot impact primary or secondary mission objectives are classified as Class D software. Typical administrative software development tools (e.g., requirements management, design, test, integration, CM, and documentation) are classified as Class E software. The QA requirements of NPR 7150.2A depend on the safety criticality designation and the class of software. NPR 7150.2A requires that software tools used to develop or maintain Class A, B, C, or safety-critical software be validated and accredited by defining acceptance processes for the tools. However, the activities and tasks comprising the acceptance processes are not specified in NPR 7150.2A. Therefore, the software tool validation and accreditation process must follow the requirements and activities of NASA-STD-8719.13C. NPR 7150.2A also requires that software tools used in the analysis, modeling, or simulation of flight software or equipment be verified, validated, and accredited in accordance with NASA-STD-7009.

NASA-STD-8739.8 requires that the QA program document tools, techniques, and methods used in software development, operation, and maintenance activities. NASA-STD-8739.8 also requires that QA personnel be trained on the software engineering design tools. NASA-STD-8739.8 contains the process (i.e., software safety litmus test) used to determine the safety criticality of software as required by NPR 7150.2A but the process is not written to accommodate the safety classification of software tools. NASA-STD-8739.8, Appendix C, contains a requirements compliance matrix that is a valuable review and approval tool. The matrix allows the reviewer to document full compliance, partial compliance, or non-compliance

with each NASA-STD-8739.8 requirement applicable to all software development processes from conception through retirement. The matrix is not currently applicable to software tools; however, the concept can be adapted for that purpose.

NASA-STD-8719.13C is a software safety standard that defines the requirements and activities for software that is created, acquired, or maintained by or for NASA, including COTS software. This standard only applies to software that is safety-critical and to software tools used to develop, test, verify, or validate safety-critical software. In accordance with this standard, software tools are subject to the same QA requirements as COTS software, including verifying the tool inputs and outputs, checking the configuration and version control of the tool, checking the limits of the tool's abilities, and addressing the potential risks of tools not performing as expected.

The NASA standards, guidebooks, and handbooks identify and recommend tools for many of the software engineering processes. NASA standards also define QA requirements for software tools. Based on the analysis of NASA documents, the following essential elements of NASA documents relative to software tools have been determined.

- (1) Software written to support development activities (e.g., verification tools and simulations) are classified separately from the system under development.
- (2) Software shall be independently classified by a QA organization and compared to the classification assigned by the project.
- (3) Software is classified as safety critical or nonsafety critical based on their safety implications to the project using the software safety litmus test of NASA-STD-8739.8.
- (4) The use of static analysis tools and other software implementation process tools should be encouraged.
- (5) Software tools used to develop or maintain Class A, B, C, or safety-critical software shall be validated and accredited by determining and defining acceptance processes.
- (6) Appendix C of NASA-STD-8739.8 is a requirements compliance matrix which lists all the requirements of NASA-STD-8739.8 and identifies the title or position of the person who has responsibility for satisfying the requirement.
- (7) Appendix F of NASA-STD-8719.13C implies that COTS tools should be treated separately from COTS software since Section F.2 addresses COTS software safety considerations and Section F.3 addresses software tools.
- (8) Software tools bought for use in testing; modeling; simulating test scenarios; configuration managing critical data and files; designing hardware; assessing flight, launch, and landing re-entry algorithms; and testing and creating configuration files for a programmable logic device will be looked at as COTS

first and foremost and subjected to the black-box COTS level of software safety assurance that involves verifying the inputs and outputs of the tool, checking the configuration and version control of the tool, checking the limits of the tools' abilities, and addressing the potential risks of tools not performing as advertised or as programmed.

- (9) NASA considers that the use of automatically generated code is in its infancy and when the code is safety-critical, or resides in an unprotected partition with safety-critical code, the automatically generated code should be subjected to the same rigorous inspection, analysis, and test as hand-generated code if the automatically generated code is accessible.

Based on the analysis of NASA documents, ISL has made the following important observations and recommendations.

- (1) Independent classification of software tools is unnecessary because tools typically have a limited functionality and are relatively easy to classify correctly; whereas application software is typically complex and multifunctional.
- (2) The NRC consider regulatory guidance for using, verifying, validating, and approving static analysis and automated testing tools.
- (3) The NRC should consider regulatory guidance for models, simulations, and analysis tools through endorsement of NASA-STD-7009.
- (4) Proper configuration control and documentation of software tools is standard industry practice and current regulatory guidance adequately addresses this practice.
- (5) The software safety litmus test should be modified and applied to software tools to better determine the level of software assurance required for the tool.
- (6) The NRC should consider that adequate software tool training, not only of software engineering staff, but of also QA personnel, be prescribed or at least designated as an acceptance criterion in pertinent regulatory guidance.
- (7) The NRC should consider the use of a requirements compliance matrix as a review and approval aid to formally document compliance to software tool requirements.
- (8) The NRC should consider regulatory guidance that clarifies the distinction between COTS software that becomes part of the final software product and COTS tools that are used in the software development process.
- (9) The NRC should consider regulatory guidance that requires automatically generated code to be scrutinized with the same rigor as hand-generated code.

2.2 Civil Aviation Industry Analysis

Aviation industry leaders believed that the airplane could not reach its full commercial potential without federal action to improve and maintain safety standards. The Air Commerce Act of 1926 charged the Secretary of Commerce with fostering air commerce, issuing and enforcing air traffic rules, licensing pilots, certifying aircraft, establishing airways, and operating and maintaining aids to air navigation. An independent Civil Aeronautics Authority was established in 1938 that conducted accident investigations, recommended ways of preventing accidents, regulated air fares, and determined routes individual carriers served. In 1958 the Federal Aviation Act transferred responsibility of commercial air standardization and oversight to the Federal Aviation Agency which conducted operations until 1967. At that time, the Department of Transportation was created and the Federal Aviation Agency was renamed the Federal Aviation Administration (FAA). FAA responsibilities increased throughout the years with concerns about pollution, noise, air traffic control system modernization, and certification of new airlines.

Throughout its history, the FAA has strived to ensure that it carries out its responsibilities by considering input from independent, outside agencies including the Radio Technical Commission for Aeronautics (RTCA). The RTCA is a Federal advisory committee whose role in aviation is to provide a forum for cooperation between government and industry on concepts of operations along with standards and guidelines for implementing new systems in aircraft. RTCA products support continuing evolution of the United States aviation system through comprehensive, industry-endorsed recommendations on issues ranging from technical performance standards to operational concepts for air transportation.

2.2.1 Document Analysis

Title 14 of the Code of Federal Regulations governs the civil aviation industry. The FAA issues standards, advisory circulars, airworthiness directives, orders, notices, handbooks, and manuals that contain requirements and guidance for complying with the federal regulations. The RTCA publishes standards and guidance material to address technical topics.

FAA-STD-026A, FAA Advisory Circular (AC) 20-115C, and FAA Order 8110.49 Change 1 are the latest FAA documents that discuss software tools. These documents support two RTCA documents, RTCA DO-178C and RTCA DO-330. The following paragraphs discuss the analysis of these five documents relative to the use, verification, validation, qualification, certification, review, and approval of software tools.

2.2.1.1 *FAA-STD-026A*

FAA-STD-026A is an older but active standard that establishes a high-level set of processes and documentation guidance for software development projects. The standard establishes and communicates requirements to be applied during the development and support of computer software. This standard references superseded versions of IEEE/EIA Standards 12207.0-1996, 12207.1-1997, and 12207.2-1997 which discuss software life cycle processes, life cycle data, and implementation considerations and represents an FAA approved tailoring of the IEEE/EIA

12207 series of standards. The applicability of this standard is primarily to software development with only a few requirements directly relevant to software tools.

Section 1.3 of FAA-STD-026A states that the requirements of the standard apply to software tools developed as an essential component of performing contract requirements whether developed internal or external to the FAA. Section 4.4 of FAA-STD-026A states that all software not developed under contract, including tools and COTS software, is regarded as nondevelopmental software. The standard contains two high-level requirements that are relevant to nondevelopmental software, including software tools. The first requirement states that all software tools need to be identified and addressed within required software development activities and documentation. The second requirement states that tools need to be validated to ensure that the tool performs its required functions and operates correctly within the software system design.

The way software tools are identified and addressed are through documentation requirements. Appendix A of FAA-STD-026A identifies 17 primary and 8 supplemental documents required to be prepared during software development. Each document is called a data item description (DID) that contains the format and content preparation instructions for data generated during the software development process. These DIDs contain complete identification and description of software and associated documentation needed to support the deliverable software including CASE tools, data in CASE tools, compilers, test tools, test data, simulations, emulations, utilities, CM tools, databases and data files, and other software. The description must include specific tool names, identification numbers, version numbers, release numbers, configurations, rationale for selection, references to user manuals, information about vendor support, and security and privacy limitations. Other than requiring the identification and description of tools, this standard does not provide any guidance on the use, verification, validation, qualification, review, or approval of tools used in a software development project.

2.2.1.2 RTCA DO-178C

RTCA DO-178C is an update to RTCA DO-178B and provides additional guidance on tool qualification because advances in software development methodologies were not adequately addressed by RTCA DO-178B. The biggest changes in RTCA DO-178C involve the criteria used to determine the qualification requirements of tools and moving tool qualification processes, activities, and tasks into separate documents. Whereas RTCA DO-178B only discusses the qualification requirements of two types of tools (i.e., development tools and verification tools), RTCA DO-178C expands the guidance by providing a method to first determine the type of tool that then is used along with the classification level of the software created with the tool to determine the qualification requirements. As in RTCA DO-178B, RTCA DO-178C still only requires tool qualification if it has not been verified and it eliminates, reduces, or automates a software development life cycle process.

RTCA DO-178C provides guidance for the production of software for airborne systems that performs its intended functions with a level of confidence in safety that complies with airworthiness requirements. Section 1 describes the purpose and scope of RTCA DO-178C and does not discuss software tools. Section 2 discusses software levels and the relationship

between the software level and the software tool level which is used in Section 12 to determine the tool qualification level (TQL). Section 3 discusses software life cycle processes, software life cycle definition, and transition criteria between software life cycle processes but does not discuss software tools. Sections 4 through 7 discuss tools in the context of the software planning, development, verification, and SCM processes. Sections 8 through 10 discuss the objectives and activities of the software quality assurance, certification liaison, and overall certification processes but do not discuss software tools. Section 11 discusses tools with respect to proper documentation of intended tool use in various plans. Section 12 discusses tool qualification requirements. RTCA DO-178C can be considered a significant source of information and guidance on the use of tools during the development of software in the civil aviation industry, covering tool planning, documentation, verification, qualification, and use in software projects. The essential features from each section that discusses software tools are discussed in the following paragraphs.

Section 2.3 of RTCA DO-178C discusses the system safety assessment process, failure conditions, software level definitions, and software level determination. The software level determination is relevant to software tools because the software level along with the type of tool as determined in Section 12.2 is used to determine the tool qualification level. As described in this section, the system safety assessment process determines and categorizes the failure conditions of the system and defines safety-related requirements to ensure the integrity of the system by specifying the immunity from, and system responses to, these failure conditions. The actual safety assessment process is not described in RTCA DO-178C. The applicant establishes the system safety assessment process to be used based on certification authority guidance which is discussed in FAA Order 8110.49 with Change 1. The software level of a software component is based on establishing how an error in a software component relates to the system failure condition(s) and the severity of the failure condition(s). The software level establishes the rigor necessary to demonstrate compliance with RTCA DO-178C and the rigor necessary to qualify tools used in developing the software. RTCA DO-178C recognizes five software levels (i.e., Level A through Level E). An ISL observation, based on an analysis of Section 2.3 of RTCA DO-178C, is that the software level of a tool is consistent with the software level of the software component created with the tool based upon the failure condition that may result from anomalous behavior of the software as determined by the system safety assessment.

- Level A: Software whose anomalous behavior would cause or contribute to a failure of system function resulting in a catastrophic failure condition for the aircraft which would result in multiple fatalities, usually with the loss of the airplane.
- Level B: Software whose anomalous behavior would cause or contribute to a failure of system function resulting in a hazardous failure condition for the aircraft which would reduce the capability of the airplane or the ability of the flight crew to cope with adverse operating conditions to the extent that there would be:

1. a large reduction in safety margins or functional capabilities,
2. physical distress or excessive workload such that the flight crew cannot be relied upon to perform their tasks accurately or completely, or
3. serious or fatal injury to a relatively small number of the occupants other than the flight crew.

Level C: Software whose anomalous behavior would cause or contribute to a failure of system function resulting in a major failure condition for the aircraft which would reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions to the extent that there would be, for example, a significant reduction in safety margins or functional capabilities, a significant increase in crew workload or in conditions impairing crew efficiency, or discomfort to the flight crew, or physical distress to passengers or cabin crew, possibly including injuries.

Level D: Software whose anomalous behavior would cause or contribute to a failure of system function resulting in a minor failure condition for the aircraft which would not significantly reduce airplane safety, and which involve crew actions that are well within their capabilities including, for example, a slight reduction in safety margins or functional capabilities, a slight increase in crew workload, such as routine flight plan changes, or some physical discomfort to passengers or cabin crew.

Level E: Software whose anomalous behavior would cause or contribute to a failure of system function with no effect on aircraft operational capability, safety, or crew workload. If a software component is determined to be Level E and this is confirmed by the certification authority, no further guidance contained in RTCA DO-178C applies and tools used to develop Level E software do not require qualification.

Section 2.4 of RTCA DO-178C discusses the architectural strategies of partitioning, multiple-version dissimilar software, and safety monitoring that may limit the impact of failures, or detect failures and provide acceptable system responses to contain them. Partitioning is a technique for providing isolation between software components to contain and isolate faults and potentially reduce the effort of the software verification process and tool qualification process. Partitioning may be achieved by allocating unique hardware resources to each software component or separating software into multiple components that run on the same hardware platform. Multiple-version dissimilar software, or diversity in the nuclear industry, is a design technique that involves producing two or more components of software that provide the same function in a way that may avoid some sources of common mode errors between components. Implementation of multiple-version dissimilar software involves a monitoring system that detects actual errors using comparator differences greater than threshold limits. Safety monitoring is a means of protecting against specific failure conditions by directly monitoring a function for failures that

would result in failure conditions. An ISL observation, based on our analysis of these architectural strategies and their relevance to software tools, is that partitioning, multiple-version dissimilar software, and safety monitoring may change the software level of the software component and the assigned level of the tool used to develop the software component and reduce the amount of rigor necessary to qualify the tool and verify the software.

Section 4.4 of RTCA DO-178C discusses the objectives and activities of the software planning process. This process produces the software plans and standards that direct the software development processes and the integral processes. Adequate software planning includes choosing a good software development environment, choosing a programming language and compiler, and choosing a software test environment. The planning process is extremely important for defining the methods, tools, and programming languages that will be used to produce the software product. The basic principal is to choose design methods, tools, and programming languages that limit the opportunity for introducing errors, and verification methods that ensure that errors introduced are detected.

With respect to planning of the software development environment, the use of tools or combinations of tools as parts of the software development environment should be chosen to achieve the necessary level of confidence that an error introduced by one part would be detected by another. An acceptable environment is produced when both parts are consistently used together. If certification credit is sought for use of the tools in combination, the sequence of the options should be examined and specified in the appropriate plan. If optional features of software tools are chosen for use in a project, the effects of the options should be examined and specified in the appropriate plan. This is especially important for compilers and autocode generators. Known tool problems and limitations should be assessed and those issues which can adversely affect airborne software should be addressed.

With respect to language and compiler considerations, adequate planning is important to support the software verification process. Using optimization when compiling source code is acceptable provided that proper planning is performed to determine the impact of the optimization. If the software planning has determined that the planned software test cases give coverage consistent with the software level, the correctness of the optimization need not be verified. Otherwise, the impact of the optimization on structural coverage analysis should be determined during the planning process. Also, since the implementation of certain compiler features may produce object code that is not directly traceable to the source code (e.g., initialization, built-in error detection, or exception handling), the software planning process should provide a means to detect this object code, provide a means to ensure verification coverage, and define the means in the appropriate plan. If a new compiler, linkage editor, or loader version is introduced, or compiler options are changed during the software life cycle, previous tests and coverage analyses may no longer be valid. The verification planning should provide a means of re-verification. Although a compiler is considered acceptable once all the verification objectives are satisfied, the compiler is only considered acceptable for that product and not necessarily for other products.

With respect to software test environment planning, the planning process defines the methods, tools, procedures, and hardware that will be used to test the outputs of the integration process. Testing may be performed using the target computer, a target computer emulator, or a host computer simulator. The planning process should determine whether the emulator or simulator needs to be qualified and that the differences between the target computer and the emulator or simulator, and the effects of these differences on the ability to detect errors and verify functionality, should be considered.

An ISL observation, based on an analysis of the software planning process, is that the essential elements of planning are to choose design methods, tools, and programming languages that limit the opportunity for introducing errors, and verification methods that ensure that errors introduced are detected.

Section 5 of RTCA DO-178C discusses the objectives and activities of the software development processes including the software requirements process, the software design process, the software coding process, and the software integration process. Section 5 of RTCA DO-178C includes some minimal guidance for tool use in the software design process related to user-modifiable software and for tool use in the software coding process related to autocode generators. Specifically, when user-modifiable software is included in the project, the non-modifiable component should be protected from the modifiable component to prevent interference in the safe operation of the non-modifiable component. This protection can be enforced by hardware, by software, by the tools used to make the change, or by a combination of the three. If the protection is provided by a tool, the tool should be categorized and qualified. Also, the use of autocode generators should conform to the constraints defined in the planning process.

Section 6 of RTCA DO-178C discusses the objectives and activities of the software verification process. The purpose of the software verification process is to detect and report errors that may have been introduced during the software development processes. The only guidance for tool use in the verification process is an acknowledgment that a tool may be used to achieve equivalence to the human verification activity. The other relevant part of the verification process to software tools is that if a tool is verified using the verification process defined in Section 6 of RTCA DO-178C, then the tool does not need to be qualified as discussed in Section 12 of RTCA DO-178C. An ISL observation, based on an analysis of the software verification process, is, with respect to tools, that source code review and analysis, verification of low-level requirements used to create the source code, verification of the software architecture, and verification of the degree of test coverage may not be possible due to the proprietary nature of the tool development if the tool is a commercially available product. If the tool cannot be fully verified using the verification process in Section 6 of RTCA DO-178C, then it must be qualified in accordance with Section 12 of RTCA DO-178C.

Section 7 of RTCA DO-178C discusses the objectives and activities of the SCM process. The SCM process ensures that configuration items are archived, recoverable, and controlled and provides a defined and controlled configuration of the software throughout the software life cycle. The objectives of SCM are independent of software level; however, RTCA DO-178C

defines two categories of software life cycle data which define the minimum SCM activities applied to the data. Control category 1 requires all SCM process activities to be applied to software life cycle data. Control category 2 is less rigorous and only requires a subset of all SCM process activities to be applied to software life cycle data. Annex A of RTCA DO-178C specifies the control category by software level for software life cycle data items.

Section 7.5 of RTCA DO-178C specifically addresses SCM processes for software tools. Configuration identification should be established for all tools used to develop, control, build, verify, and load the software. If tools have not been qualified, then the SCM processes consistent with control category 2, as a minimum, should be satisfied. The SCM processes for qualified tools are defined in RTCA DO-330. Generally, the SCM processes for qualified tools should comply with control category 1 for software development tools and control category 2 for verification tools.

Section 11 of RTCA DO-178C discusses the characteristics, form, CM controls, and content of the software life cycle data and contains some general guidance to support the certification of software tools. This section states that the plan for software aspects of certification is the primary means used by the certification authority for determining whether an applicant is proposing a software life cycle that is commensurate with the rigor required for the level of software being developed. This plan should include a description of specific considerations that may affect the certification process including tool qualification.

Section 11 of RTCA DO-178C also states that the software development plan, the software verification plan, the SCM plan, and the software QA plan should identify methods and tools to be used, the coding methods, programming language, coding tools to be used, and the hardware platforms for the tools to be used. The software verification plan should also include a description of the assumptions made by the applicant about the correctness of the compiler, linkage editor, or loader. Section 11 of RTCA DO-178C also states that the software requirements standards, the software design standards, and the software coding standards should include any constraints on the use of tools used for requirements development, design, and coding.

Sections 11.15 and 11.16 of RTCA DO-178C discuss the software life cycle environment configuration index and the software configuration index. The software life cycle environment configuration index identifies the configuration of the software life cycle environment and should identify the tools to be used during the development of the software, identify the software testing and analysis tools, and identify qualified tools and their associated tool qualification data. The software configuration index identifies the configuration of the software product and should identify procedures, methods, and tools for making modifications to the user-modifiable software, if any, and should identify instructions for building the executable object code including instructions for compiling and linking.

An ISL observation, based on an analysis of Section 11 of RTCA DO-178C, is that tool use should be planned in advance and documented in the appropriate software development, verification, CM, and QA plans to support the certification process. The requirements of Section

11 of RTCA DO-178C are consistent with the requirements of FAA-STD-026A which require adequate identification and descriptions of all tools used in the software development process.

Section 12 of RTCA DO-178C provides guidance on using tools to modify previously developed software, tool qualification, and using tools to develop multiple-version dissimilar software. With respect to using or modifying previously developed software, any change in the development environment, the target processor or other hardware, or integration with software other than that used for the original application requires special consideration. If a new development environment uses software tools, the tool may need to be re-qualified. Using a different autocode generator or a different set of autocode generator options may change the source code; therefore, the impact of these changes should be analyzed. If a different compiler or options are used, previous verification may not be valid and should not be used for the new application. An ISL observation, based on an analysis of the guidance for previously developed software, is that tools used to develop software used in a previous application must be re-verified or re-qualified if any aspect of the development environment, hardware, or related software has changed.

The most important guidance for software tools in this document is related to tool qualification requirements. A software tool needs to be qualified when:

- (1) the tool eliminates, reduces, or automates some software development life cycle process,
- (2) the software tool output has not been verified, and
- (3) the tool is used to develop software that has some impact on the aircraft operational capability, safety, or crew workload.

Tools that do not eliminate, reduce, or automate software development life cycle processes do not need to be qualified. Tools that eliminate, reduce, or automate life cycle processes only need to be qualified if the tool output cannot be verified. Tools used to develop software whose anomalous behavior would cause or contribute to a failure of system function with no effect on aircraft operational capability, safety, or crew workload do not need to be qualified. Most, if not all, COTS software tools need to be qualified because they typically automate a life cycle process and verification is usually not possible given the proprietary and unmonitored nature of the software tool development environment.

If a tool needs to be qualified, the amount of rigor necessary to qualify the tool is determined through the assignment of a tool qualification level (TQL) which depends on two important factors. The first factor is the type of tool (i.e., the impact of tool use in the software life cycle processes). The second factor is the safety level of the software which is being developed using the tool as determined by the system safety assessment. The determination of the TQL is deterministic and not based on risk or probability. The software level is based on a safety assessment which assigns a safety level based on a deterministic assessment of system failures caused by software errors. The type of tool and its impact on the software life cycle processes is determined using the following deterministic criteria.

- Criterion 1: A tool whose output is part of the airborne software and thus could insert an error.
- Criterion 2: A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of:
- a. Verification process(es) other than that automated by the tool, or
 - b. Development process(es) that could have an impact on the airborne software.
- Criterion 3: A tool that, within the scope of its intended use, could fail to detect an error.

A strict interpretation of Criterion 1 would limit tools in this category to autocode generators, compilers, and linkers which directly generate output that becomes part of the final executable airborne software. However, as discussed in Appendix D of RTCA DO-330, Criterion 1 actually covers all software tools used in any software development life cycle process from generating high-level requirements, design requirements, low-level requirements, source code, executable object code, data files, and configuration files. Criterion 1 covers all development tools previously covered by RTCA DO-178B that can introduce an error into the executable object code at any point in the development process. Criterion 2 is new compared to RTCA DO-178B and covers verification tools that not only automate verification processes but whose output could eliminate or reduce the need to complete other verification or development processes. Criterion 3 is the same as the verification tool classification in RTCA DO-178B and covers verification tools that only reduce, eliminate, or automate verification processes.

Based on the impact of a tool on the software life cycle processes as specified by one of the above criteria and the software level as determined by the system safety assessment that assesses the safety implications of a failure in the software, the TQL is determined from the following table. TQL-1 is the most rigorous level and TQL-5 is the least rigorous level. The objectives, activities, guidance, and life cycle data required for each TQL to qualify the software tool are described in RTCA DO-330. An ISL observation, based on an analysis of tool qualification requirements, is that not all tools need to be qualified but if they do, the amount of rigor necessary to qualify the tool depends on both the type of tool and the safety level of the software that the tool is being used to develop. ISL recommends that the NRC consider adopting a similar process for commercial nuclear power that would determine tool qualification requirements based on a two-dimensional, deterministic process of assigning a software level based on a safety assessment of the system containing the software and performing a tool impact assessment to determine the type of tool and the tool's impact on the software development life cycle processes.

Table 2.1 – Tool Qualification Level Determination

Software Level	Criterion		
	1	2	3
A	TQL-1	TQL-4	TQL-5
B	TQL-2	TQL-4	TQL-5
C	TQL-3	TQL-5	TQL-5
D	TQL-4	TQL-5	TQL-5
E	No tool qualification necessary		

The final content of Section 12 of RTCA DO-178C provides guidance on tools used to develop multiple-version dissimilar software or the use of dissimilar simulators for software verification and testing. If multiple-version dissimilar software is used, the tool qualification process can be modified, if evidence is available that the multiple software tools used in the software development process are dissimilar. Evidence should show that each tool was obtained from a different developer and show that each tool has a dissimilar design. If the evidence is provided, some of the qualification processes can be eliminated. If separate, dissimilar simulators are used to verify multiple-version dissimilar software versions, then the approach to tool qualification of the simulators may be modified. Unless it can be justified as unnecessary for multiple simulators to be dissimilar, evidence should be provided that shows that each simulator was developed by a different team, each simulator has different requirements, a different design, a different programming language, and each simulator executes on a different processor. An ISL observation, based on an analysis of multiple-version dissimilar software, is that qualification activities can be modified from the activities specified in RTCA DO-178C by providing evidence of the dissimilar nature of multiple-version dissimilar software and dissimilar simulators.

2.2.1.3 RTCA DO-330

RTCA DO-330 presents an entire qualification and development process for software tools that mirrors the software development processes described in RTCA DO-178C, tailored for software tool development. Analogous to software development, the software tool development guidance provided in RTCA DO-330 recommends the application of a life cycle process with associated objectives and activities to meet those objectives. The objectives and activities that apply to a specific tool development project vary depending on the tool qualification level (TQL) assigned in RTCA DO-178C. As discussed in the previous section, RTCA DO-178C identifies five tool qualification levels (TQL-1 through TQL-5). TQL-1 is the most rigorous qualification level and requires well defined and executed tool development, verification, and integral processes with the highest degree of verification independence because the tool can introduce an error into the software and a failure of that software can cause a catastrophic failure of the aircraft. The remaining TQLs require decreasing amounts of rigor. TQL-5 is the least rigorous qualification level because the tool cannot introduce an error into the software and can only fail to detect a software defect. An ISL concern, based on our analysis of the software tool qualification requirements, is that the rigor of qualification of TQL-5 tools should be increased because verification software classified as TQL-5 can fail to detect software defects that can

potentially cause a catastrophic failure of the aircraft. The level of rigor associated with the qualification of TQL-5 verification tools may be appropriate for level C or D software; however, more qualification rigor should be applied to verification tools associated with level A or B software.

The qualification process presented in RTCA DO-330 contains a tool qualification planning process, tool operational requirements definition process, tool development processes, tool operational integration process, and tool development support processes. The tool development support processes include a tool verification process, a tool CM process, a tool QA process, and a tool qualification liaison process. All of the tool qualification processes described in RTCA DO-330 apply to software tools being developed as part of a software project, either in-house or through a specific outside vendor or contractor, where the responsibility of tool qualification is shared by both the tool developer and the tool user. Only a limited number of the tool qualification processes apply to COTS tools because the tool has already been commercially developed and made available to the public for general use. An ISL observation, based on an analysis of the tool qualification process, is that a tool is developed and verified using a life cycle process with objectives and activities similar to a standard software development life cycle process with all the associated planning, QA, and documentation requirements. A tool is considered qualified when all the tool development life cycle process objectives and activities have been completed.

Section 4 of RTCA DO-330 describes the tool qualification planning process that mirrors the software planning process described in Section 4 of RTCA DO-178C except it has been tailored for software tool development in lieu of software development. The tool qualification planning process defines one or more life cycles by choosing the activities for each process, specifying a sequence for the activities, and assigning responsibilities for the activities. The tool life cycle processes include the tool planning process, the tool development processes that produce the tool, and the integral processes that ensure the correctness, control, and confidence of the tool life cycle processes and their outputs.

Sections 4.3 and 4.4 of RTCA DO-330 describe the objectives and activities of the tool planning process that defines and coordinates the activities of tool development and integral processes. The tool planning process objectives and activities are almost identical to the RTCA DO-178C software planning process objectives and activities except for the tailoring for tool planning. The tool planning process objectives and activities applicable to a given tool qualification are determined by the TQL and specified in RTCA DO-330, Annex A, Tables T-0 and T-1. Satisfaction of the objectives of the tool planning process are discretionary for TQL-5 tools. Most tool planning process objectives are also discretionary for tools characterized as TQL-4 and all objectives should be satisfied for TQL-1 through TQL-3 tools. One unique aspect of the tool planning process is that any tool used to develop another tool is subject to the same qualification requirements as the tool being developed. For a tool that can introduce an error in the outputs of a tool, the applicable TQL is the same as the tool being developed. For a tool that cannot introduce an error in the output of the tool, but may fail to detect an error in the tool life cycle data, the applicable TQL is TQL-5. Documents generated during the tool planning process include the tool qualification plan, tool development plan, tool verification plan, tool CM

plan, and the tool QA plan. ISL analysis of the tool planning process of RTCA DO-330 found that the documents that must be generated are identical to the documents required to be generated in the software planning process discussed in Section 4.3 of RTCA DO-178C, except the tool qualification plan replaces the plan for software aspects of certification. An ISL observation, based on our analysis, is that the development of a software tool is planned in the same manner as the development of software and that the tool planning process only applies to tools that are being developed along with or specifically for the software and does not apply to COTS tools.

Section 5 of RTCA DO-330 discusses the objectives and activities of the tool development life cycle and associated processes. The overall tool development life cycle includes three major processes: the tool operational requirements definition process, the tool development processes, and the tool operational integration process.

The tool operational requirements definition process is a unique tool qualification specific process. The tool operational requirements definition process generates the tool operational requirements from the software plans and tool plans developed in Section 4 of RTCA DO-330. The tool operational requirements identify how the tool is to be used within the software life cycle processes and should include enough detail to demonstrate that the functionality and the outputs of the tool correspond to the identified software life cycle activities to be performed by the tool. Table T-0 in Annex A of RTCA DO-330 summarizes the objectives, activities, and output of the tool operational requirements process by TQL.

The tool development processes include the tool requirements process, tool design process, tool coding process, and tool integration process. These processes are similar to the software development processes of RTCA DO-178C, Section 5, except for some minor tailoring for applicability to tools. These processes transform the tool operational requirements into high-level tool requirements, develop the tool architecture and low-level tool requirements, develop tool source code from the low-level tool requirements, and produce the tool in an executable format. Table T-2 in Annex A of RTCA DO-330 contains a summary of the objectives and outputs of the tool development processes by TQL. All objectives should be satisfied for tools characterized as TQL-1, TQL-2, and TQL-3. Some objectives should be satisfied for TQL-4 tools and no tool development process objectives need to be satisfied for TQL-5 tools.

The final tool development life cycle process is the tool operational integration process which is a unique tool qualification specific process. The purpose of the tool operational integration process is to install the tool executable object code in the tool operational environment and to gain confidence that the tool is operating as expected. Table T-0 in Annex A of RTCA DO-330 summarizes the objectives, activities, and output of the tool operational integration process by TQL. An ISL observation, based on our analysis of the tool development process objectives, is that some of the objectives for TQL-4 and TQL-5 tools that are identified as discretionary in Annex A, Table T-2 of RTCA DO-330, should be required to be satisfied. The three objectives that should be satisfied for TQL-5 tools are the development of tool source code, producing the executable object code, and installing the executable object code in the verification environment. It is not clear why Table T-2 of RTCA DO-330 identifies these objectives as

discretionary because without these three objectives being satisfied, the tool cannot be executed. For COTS tools, developing the source code and producing the executable object code is not necessary; however, COTS tools require installation of an executable in a verification environment; therefore, the third objective identified above should not be discretionary for COTS tools. Another ISL observation is that the categorization of some of the tool development objectives as discretionary should be reconsidered for TQL-4 and TQL-5 tools if the tool qualification objectives and activities from this document are considered for developing new tool qualification regulatory guidance.

Section 6 of RTCA DO-330 discusses the objectives and describes the activities for a two-part overall tool verification process. The first part of the overall tool verification process is the tool verification process. The tool verification process mirrors the software verification process of RTCA DO-178C, Section 6. The second part of the overall tool verification process is the tool operational V&V process which is unique to tool qualification and provides confidence that the tool functionality and outputs comply with the software life cycle needs.

The tool verification process, the first part of the overall tool verification process, is the technical assessment of the outputs of both the tool development processes and the tool verification process. The purpose of the tool verification process is to detect and report errors that may have been introduced during the tool development processes. Tool verification objectives are satisfied through a combination of reviews, analyses, development of test cases and procedures, and the subsequent execution of those test procedures. Annex A, Tables T-3 through T-7 of RTCA DO-330, contain a summary of the objectives and outputs of the tool verification process by TQL.

The tool operational V&V process, the second part of the overall tool verification process, provides confidence that the outputs and the functionality of the tool comply with the software life cycle process needs. The verification objectives of the tool operational V&V process consist of the detection and reporting of errors that may have been introduced during the tool development processes by verifying that tool operational requirements are complete, accurate, verifiable, and consistent and by verifying that the functionality and the outputs of the tool installed in the tool operational environment comply with the tool operational requirements. Simply speaking, the verification objectives ensure that the tool has been built correctly. The validation objectives of the tool operational V&V process consist of the analysis of the functionality and the outputs of the tools for correctness and completeness with respect to the software life cycle activities performed. Simply speaking, the validation objectives ensure that the correct tool has been developed by ensuring that the tool meets the needs of the software life cycle process that is being eliminated, reduced, or automated. Annex A, Table T-0 of RTCA DO-330, contains a summary of the objectives and outputs of the tool operational V&V process.

An ISL concern, based on our analysis of the overall tool verification process, is that objectives of the verification process are discretionary for TQL-5 tools and only some of the objectives need to be satisfied for TQL-4 tools. TQL-4 and TQL-5 tools are tools used in the software verification process that may fail to detect an error in software classified as Level A through Level D and may eliminate, reduce, or automate other software development life cycle

processes in addition to the verification process eliminated, reduced, or automated directly by the tool. Categorizing all of the verification objectives for a TQL-5 tool as discretionary is a concern. As a minimum, some tool testing activities should be performed for TQL-5 tools, especially verifying that the test results are correct and discrepancies explained. Also, all of the verification objectives for the output of the coding and integration process are discretionary for TQL-4 tools. As a minimum, activities for TQL-4 tools that ensure that the tool source code is verifiable and that the output of the tool integration process is complete and correct should be mandatory and not discretionary. Any regulatory guidance based on RTCA DO-330 for tool qualification should reconsider the verification process objectives that need to be satisfied for TQL-4 and TQL-5 tools.

Section 7 of RTCA DO-330 discusses the objectives and describes the activities of the tool CM process which primarily provides a defined and controlled configuration of the tool throughout the tool life cycle and ensures secure physical archiving, recovery, and control for configuration controlled items. Configuration management activities include configuration identification; baselines and traceability; problem reporting and tracking; change control; change review; configuration status accounting; and archive, retrieval, and release. The tool CM process mirrors the SCM process outlined in RTCA DO-178C, Section 7, except for minor tailoring for applicability to tools. Annex A, Table T-8, contains a summary of the tool CM objectives. Currently, all objectives should be satisfied for TQL-1 through TQL-4 tools; however, only two objectives need to be satisfied for TQL-5 tools. Those two objectives are that configuration items need to be identified and archive, retrieval, and release capabilities need to be established. An ISL observation, based on our analysis of tool CM, is that tools classified as TQL-5 should also be required to satisfy the objective of establishing problem reporting, change control and review, and configuration status accounting with emphasis primarily on problem reporting. Problem reporting and resolution are important for verification tools to track which software errors have not been found by the tool so that the problem can be cycled back to the tool developer. Any regulatory guidance based on RTCA DO-330 for tool qualification should reconsider the CM objectives that need to be satisfied for TQL-5 tools.

Section 8 of RTCA DO-330 discusses the objectives and activities of the tool QA process to assure that objectives are satisfied; deficiencies are detected, evaluated, tracked, and resolved; and the tool product and tool life cycle data conform to stated requirements. Annex A, Table T-9, contains a summary of the tool QA process objectives by TQL. An ISL observation, based on our analysis of the tool qualification process, is that the tool QA process mirrors the software QA process outlined in RTCA DO-178C, Section 8, except for minor tailoring for applicability to tools.

Section 9 of RTCA DO-330 discusses the tool qualification liaison process. This is not a standalone process and instead is performed within the framework of the certification liaison process of the software. Liaison with the personnel responsible for approving the tool is needed for tool qualification. The tool qualification liaison process identifies the activities that are required to obtain approval of the tool qualification. These activities are concerned with identifying the need for tool qualification, analyzing all known problems and functional limitations of the tool, and reporting the results of the analysis. Annex A, Table T-10, contains a summary

of the tool qualification liaison process activities by TQL. All objectives should be satisfied for all five TQL classifications. An ISL observation, based on our analysis of the tool qualification liaison process, is that frequent communication with the approval authority is paramount for obtaining approval of tools. Another ISL observation is that the essential elements of the tool qualification liaison process include submitting evidence that tool problems and functional limitations have been analyzed, resolved, and properly reported; the tool qualification process has been completed successfully; and all objectives that should have been satisfied in the tool qualification process have actually been satisfied. The tool qualification liaison process described in this section could form the technical basis for developing a liaison process between the NRC and the organization seeking approval of a tool.

Section 10 of RTCA DO-330 provides guidance for the typical tool qualification life cycle data produced during the tool qualification process and the minimum contents of each data item. This section does not provide any tool qualification guidance but does provide guidance for organizing and controlling the content of various plans, standards, indices, reports, records, and summaries generated during the tool qualification process. An ISL observation, based on our analysis of tool qualification life cycle data guidance, is that the essential elements of tool qualification data management are to thoroughly plan the qualification of the tool; clearly identify the functional requirements of the tool and what software development processes are being reduced, eliminated, or automated; maintain bi-directional traceability between tool requirements and tool source code; fully document the tool verification process including all test procedures and results; maintain the ability to regenerate the tool executable and all associated data files; and fully document the use and qualification basis of any other tools used to develop and qualify the tool. Another ISL observation is that the tool qualification life cycle data for TQL-5 tools is lacking in some key areas because most qualification process objectives do not have to be satisfied for these tools and therefore, no data is generated. ISL recommends that any new regulatory guidance for tool qualification based on RTCA DO-330 should consider adding more tool qualification levels or changing existing tool qualification levels in the Criteria 2 and 3 categories to require more objectives to be satisfied for tools used in the development of Level A and B software. A final ISL observation, based on our analysis of tool qualification levels, is that the number of objectives that need to be satisfied and the data required to be produced for TQL-5 tools used to develop Level A and B software is insufficient considering that the tool may fail to find a defect in software that could cause a catastrophic or hazardous failure of the aircraft.

Section 11 of RTCA DO-330 provides guidance on five additional considerations concerning tool qualification related to the use of multi-function tools, previously qualified tools, COTS tools, service history, and alternative methods for achieving compliance with objectives of RTCA DO-330. The first additional consideration provides guidance for multi-function tools. The key element associated with multi-function tools is to assign the appropriate TQL. A TQL should be determined for each function of the tool and the entire tool qualified to the highest TQL unless protection between the functions of the tool can be demonstrated. Only those functions of the tool used in the software life cycle process need to be qualified and the qualified function must not depend on the unused functions. An ISL observation, based on our analysis of the guidance for multi-function tools, is that the guidance is complete and reasonable. ISL agrees

with the statements in Section 11 of RTCA DO-330 that express concern over tools that perform both development and verification activities. The tool qualification process for these types of tools must ensure that sufficient protection techniques have been implemented to avoid an error that might affect both functions. If the verification objective being satisfied by the tool requires independence, the function that verifies the output should be shown to be independent of the function that produces the output to ensure that common errors do not exist between tool functions and to verify that the verification function completely detects any errors in the output.

The second additional consideration in Section 11 of RTCA DO-330 provides guidance for tool reuse and whether the tool needs to be re-qualified for the current project. If the tool, its operating environment, and the tool operational requirements are unchanged, and the TQL for the current project is the same or lower than the previous project, then the tool can be reused without any additional qualification. If the tool, its operating environment, or its operational requirements have changed such as through an upgrade of the operating system, a higher TQL is required, or the intended use of the tool has changed, then an impact analysis must be performed to determine any re-verification activities. An ISL observation, based on our analysis of the guidance for tool reuse, is that the guidance should be more conservative with respect to determining the impact of changes in the tool operating environment. The section does not define what constitutes a change in the tool operating environment. Even though the operating system is the same, subtle changes to the tool operating environment are possible due to the installation of other software and these changes may be overlooked. Therefore, based on our analysis, ISL considers that at least a minimal set of verification activities should be performed on the tool to confirm that subtle changes in the tool operational environment have been assessed.

The third additional consideration in Section 11 of RTCA DO-330 provides guidance for COTS tools. This section is important to the software tool research project since most tools are COTS. A COTS tool is typically developed independent of any specific software project and is intended to be utilized by multiple users. Section 11.3 of RTCA DO-330 discusses how the RTCA DO-330 guidance should be adapted for application to a COTS tool. The section states that COTS tools need to comply with the objectives appropriate to the necessary TQL assigned to the tool with tool qualification activities split between the tool developer and the tool user.

Section 11.3.2 of RTCA DO-330 states that the tool developer should perform the activities and produce the tool life cycle data necessary to demonstrate that all tool developer objectives have been satisfied. The tool developer should provide tool operational requirements so the tool user can determine if the tool meets the software life cycle needs. Typically, tool operational requirements would be in the form of a user manual or installation guide which accompanies most COTS tools. However, Section 11.3.2.1 of RTCA DO-330 states that the tool operational requirements should include, in addition to a user manual, a description of all anticipated tool operational environments, input files, and output files; requirements for all expected functional behavior of the tool; and requirements to address any abnormal activation modes or inconsistent inputs that should be detected by the tool. Sections 11.3.2.2 through 11.3.2.4 of RTCA DO-330 further require the tool developer to generate a tool qualification plan, a tool configuration index, and a tool accomplishment summary.

Section 11.3.3 of RTCA DO-330 continues with tool qualification activities that should be performed by the tool user which primarily assesses the tool in the tool operational environment. Sections 11.3.3.1 through 11.3.3.4 of RTCA DO-330 state that the tool user should generate tool operational requirements that augment the developer tool operational requirements, a tool qualification plan that defines the division of responsibilities between the tool developer and the tool user, a tool configuration index that identifies data produced by the tool user, and a tool accomplishment summary that identifies the options selected during tool use, demonstrates that the tool installation and configuration are in accordance with the developer recommendations, and contains the results of both the tool developer's and user's verification activities.

An ISL observation, based on our analysis of the COTS tool qualification objectives and activities, is that the objectives and activities require considerable effort on the part of the COTS vendor that may be difficult to obtain. Since the tool was developed by a vendor independent of any specific application and without knowledge of what TQL would be assigned to the tool, ISL considers that it is unlikely that the developer would have performed the required activities or produced the required data. ISL also considers that it is unlikely that the vendor's development process can be impacted by feedback from an organization using the COTS tool. Thus, a more practical approach to COTS tool qualification is required that does not rely on vendor participation and generation of qualification data beyond what is in the user's manual. COTS tool qualification should concentrate on the use of service history, exhaustive input testing, formal verification methods, or dissimilar tools.

The use of service history is the fourth additional consideration for tool qualification and is discussed in Section 11.4 of RTCA DO-330. The key elements of service history are the availability of service history data, adequate problem reporting and resolution, similarity of the intended tool operational environment compared to the environment in which the service history data was collected, level of expertise in the use of the tool, and the tool's change history.

Section 11.5 of RTCA DO-330 discusses potential alternative methods of tool qualification which is the fifth additional consideration for tool qualification. Potential alternative methods for tool qualification include exhaustive input testing, formal methods, and dissimilar tools. The impact of using one of these alternative methods must be specified in the tool plans including the rationale for use of the alternative methods which shows how the objectives of the tool qualification process are satisfied.

2.2.1.4 *FAA Advisory Circular 20-115C*

FAA AC 20-115C was written by the FAA to recognize RTCA DO-178C, RTCA DO-330, and RTCA DO-330 supplements RTCA DO-331 through 333 as an acceptable means for showing compliance with airworthiness regulations for software.

Review and approval of software tools is dependent on the applicant following all the objectives and developing all the associated life cycle data as required by RTCA DO-178C, RTCA DO-330, and all its supplements. The applicant must plan and execute all activities that will satisfy each objective in those documents. Specifically, the life cycle data specified in RTCA DO-178C, Section 9.3, must be submitted. For tool qualification, the life cycle data specified in RTCA DO-

330, Section 9.0.a, must be submitted to the certification office and the applicable tool qualification data described in RTCA DO-178C, Section 11, must be made available to the FAA. If any of the supplements to RTCA DO-330 are used to satisfy the RTCA DO-330 objectives, various documents must contain information on how the supplements will be used. The plan for software aspects of certification should identify which supplements apply and will be used. The tool qualification plan (TQP) should describe how RTCA DO-330 and the supplement guidance will be applied to tool development or verification.

Section 10 of FAA AC 20-115C addresses tool qualification and states that RTCA DO-178C, Section 12.2, and RTCA DO-330 and its supplements, if used, provide an acceptable method for tool qualification. This section acknowledges that RTCA DO-178C addresses tool qualification for new or modified tools; however, it does not address the use of tools previously qualified to the RTCA DO-178B criteria. FAA AC 20-115C provides a table to determine the correlation between a tool previously qualified as a RTCA DO-178B development tool or verification tool and RTCA DO-178C tool criteria. Basically, all RTCA DO-178B development tools are covered under RTCA DO-178C, Criterion 1, and assigned a TQL based on the software level developed by the tool. RTCA DO-178B verification tools that meet RTCA DO-178C, Criterion 2, are assigned to TQL-4 or TQL-5 depending on the software level. RTCA DO-178B verification tools that meet RTCA DO-178C, Criterion 3, are assigned to TQL-5.

FAA AC 20-115C is similar to an NRC regulatory guide in that it endorses the use of RTCA DO-178C, RTCA DO-330, and RTCA DO-330 supplements as a means of qualifying tools. It also provides guidance for FAA review of tool use, similar to a review plan, and provides guidance on obtaining FAA approval of tools, similar to ISG-06 or a regulatory guide that specifies what must be submitted with an application for approval by the NRC. FAA AC 20-115C encourages the use of these endorsed documents for qualifying new tools or re-qualifying modified tools. It also addresses whether previously qualified tools need to be re-qualified or whether they can be used without re-qualification. FAA AC 20-115C also describes the required objectives, activities, and data that must be submitted to facilitate review and approval of the tool. Thus, FAA AC 20-115C provides a sound technical basis upon which to develop a software tool review and approval process for commercial nuclear power.

2.2.1.5 *FAA Order 8110.49*

FAA Order 8110.49 and FAA Order 8110.49 with Change 1 clarify software review and approval processes described in RTCA DO-178B, determine the level of FAA involvement in the software review and approval process, provide assistance in approving airborne software by determining that appropriate processes and procedures have been established that result in compliance to RTCA DO-178B objectives, and discuss the qualification of software tools. FAA Order 8110.49 with Change 1 is an update to FAA Order 8110.49 to incorporate policy from FAA Notice 8110.110 to provide guidance for properly overseeing software suppliers, reporting software problems, assuring airborne system databases, and managing the software development or verification environment. The incorporation of policy from FAA Notice 8110.110 does not add any significant guidance for software tools to FAA Order 8110.49 so the analysis and discussion

of these policies is limited. The following discussion applies to FAA Order 8110.49 with Change 1.

FAA Order 8110.49 is somewhat obsolete because it discusses software tool review, approval, verification, and qualification using the methods, procedures, and processes of RTCA DO-178B and does not support the methods, procedures, and processes of RTCA DO-178C, RTCA DO-330 and supplements RTCA DO-331, RTCA DO-332, and RTCA DO-333. FAA AC 20-115C recognizes RTCA DO-178C and its supplements as an acceptable means of compliance for securing FAA approval of software in airborne systems and equipment; however, FAA Order 8110.49 does not currently provide assistance in obtaining FAA approval of airborne software using the processes of RTCA DO-178C. Nevertheless, the analysis and discussion of FAA Order 8110.49 is still relevant to the software tools project because some of the high-level guidance continues to be relevant to software tools and remains consistent with the methods, procedures, and processes of both RTCA DO-178B and RTCA DO-178C.

Section 2 of FAA Order 8110.49 clarifies the application of the certification liaison process described in Section 9 of RTCA DO-178B. The clarifications also apply to the certification liaison process described in Section 9 of RTCA DO-178C. FAA Order 8110.49 specifies that software life cycle processes and data should be reviewed as early in the software life cycle as possible and integrated throughout the software life cycle to assess compliance to RTCA DO-178B and to mitigate the risk that the system, software, and planning decisions will not satisfy RTCA DO-178B. FAA Order 8110.49 describes four types of reviews, some of which involve software tools. These four reviews are discussed in the following paragraphs.

The first review is a software planning review that is conducted when the initial software planning process is complete. The software planning review ensures that the planning process establishes various plans, standards, procedures, activities, methods, and tools required to develop, verify, control, assure, and produce the software life cycle data and to determine if the plan provides an acceptable means for satisfying the objectives of RTCA DO-178B. One of the required plans is a tool qualification plan. The content of the tool qualification plan is described in Section 12.2.3.1 of RTCA DO-178B; however, with the update to RTCA DO-178C, that content has been moved to Section 10.1.2 of RTCA DO-330. The tool qualification plan should include identification of the tool, qualification considerations including the tool qualification level, a functional overview of the tool, description of the tool operation and development environments, tool life cycle description and the qualification activities to be performed, means of providing visibility of the tool life cycle process activities so tool reviews can be planned, a means of ensuring that supplier processes and outputs comply with approved tool plans and standards if suppliers are used, and any additional considerations that may affect the qualification process such as using COTS tools. Early and frequent planning for tool use are essential elements of software development. ISL recommends that the NRC consider, in any software tool regulatory guidance, stressing the need and provide guidance for adequate planning.

The second review is a software development review that is conducted when at least 50 percent of the software development requirements, design, and code are completed and reviewed. The

software development review assesses the effective implementation of the plans and standards through examination of the life cycle data generated during the software development and integral processes. Since the verification process is a software development support process performed concurrently with all other processes, FAA Order 8110.49 specifies the review of software verification procedures and verification results as part of the software development review even though the description of the activities only cover the generation of high-level requirements, software architecture, low-level requirements, and source code. Software verification results are also reviewed as part of the software planning review, the software verification review, and the final certification review. A benefit of multiple, formal certification authority verification reviews is a higher probability of finding areas that can be improved early in the development process. With respect to software tools, FAA Order 8110.49 does not identify any software tool related reviews as part of the software development review. Since the software development process includes the use of many software tools, the development review should consist of some type of confirmation that the software tools used in this process (e.g., compilers, linkers, loaders, and autocode generators) are performing consistent with their requirements. An ISL observation, based on our analysis, is that the NRC consider including, in any new regulatory guidance, some type of tool operational review as part of the software development review even though FAA Order 8110.49 does not include this type of reviews.

The third review is a software verification review that is conducted when at least 50 percent of the software verification and testing data is complete and reviewed. The software verification review ensures that the software verification processes confirm that the software product specified is the software product built. With respect to software tools, FAA Order 8110.49 requires software tool qualification data be available for the software verification review. The tool qualification data is described in Section 12.2.3 of RTCA DO-178B; however, with the update to RTCA DO-178C, that information has been moved to Section 10 of RTCA DO-330. Tool qualification data includes information about the tool in various plans which identify the intended use of the tool and information about assessing its impact on software life cycle processes. Tool qualification data also includes tool requirements, which describe the tool functionality, and tool verification cases and results. The tool qualification data required to be available for the verification review seems relevant to software tools developed specifically for a software development project but does not seem to be appropriate for COTS tools. An ISL observation, based on an analysis of the software verification review, is that the NRC should consider, in any new regulatory guidance, expanding the software verification review to cover verification reviews of both developed and COTS tools.

The fourth review is a final certification software review that is conducted after the final software build is complete, the software verification is complete, a software conformity review has been conducted, and the software application is ready for formal system certification approval. This review does not include any reviews of software tools or software tool data.

Section 3 of FAA Order 8110.49 provides guidance for determining the FAA level of involvement in software projects. The level of FAA involvement is LOW, MEDIUM, or HIGH depending primarily on the software level (A, B, C, D, or E) as defined in Section 2.2 of RTCA DO-178B or Section 2.3 of RTCA DO-178C. The level of FAA involvement for software levels A, B, and C

also depends on other criteria such as software certification experience, software development capability, software service history, software complexity, and experience and capabilities of the FAA certification authority's designated engineering representative or person responsible for software oversight. With respect to software tools, the use of unusual tools increases the level of FAA involvement. For software level A or B projects, the use of unusual tools requires a HIGH level of FAA involvement. For software level C or D projects, the use of unusual tools requires at least a MEDIUM level of FAA involvement. An ISL observation, based on our analysis of FAA involvement in software and software tool review and approval, is that with the expanded use and continuing development of new tools used in almost all phases of software development, the level of involvement of the certification authority or the approval agency should be higher for tools other than those classified as unusual. Tool complexity, tool familiarity, tool supplier, and tool service history should all be factors to consider when determining the level of involvement. The essential element of regulatory involvement in a software development project is that any regulatory guidance addressing the level of regulatory involvement in the software development process should expand on the FAA guidelines and base the level of involvement on tool characteristics other than just whether the tool is unusual. Even a frequently used tool that is complex and is COTS software could be a policy issue and should increase the level of regulatory involvement.

Section 7-6 of FAA Order 8110.49 discusses requirements for user-modifiable software and supplements Section 5.2.3 of RTCA DO-178B and Section 5.2.3 of RTCA DO-178C. User-modifiable software (UMS) is software within a system approved for user modification. UMS does not apply to option-selectable software except where such software is also user-modifiable. Users (e.g., airlines and operators) may modify UMS within the specified modification constraints and with approved modification procedures without involvement by the certification authority. It is intended that once the system with the UMS has been certified, modification of the UMS by the user should have no effect on the aircraft safety margins, aircraft operational capabilities, flight crew workload, any non-modifiable software components, or any protection mechanisms of the system. FAA Order 8110.49, Section 7-6, requires that the non-modifiable software components be protected from modifiable components to prevent interference with the safe operation of the non-modifiable software components. To enforce this protection, tools are allowed to make changes to the modifiable component provided the following information is submitted to the certification authority for approval:

- (1) plans for controlling tool version,
- (2) plans for controlling tool usage,
- (3) plans for qualifying or verifying the tool, and
- (4) procedures for modifying the tool.

Software forming a component of the tool and used in the protective function should be developed to the software level of the most severe failure condition of the system, as determined by a system safety assessment. Use of software tools for user modifications requires tool qualification and approval of procedures to use and maintain the tool. Changes to

the tool or procedures may require re-qualification of the tool. Section 7-7 of FAA Order 8110.49 states that the plan for software aspects of certification should identify the intention to develop a system containing UMS and that if software tools are to be used for modifying UMS, the certification plans should identify tool qualification plans or verification procedures to ensure that the tool has modified the UMS to approved procedures and constraints, and it has not affected the non-modifiable software or protection mechanisms. An ISL observation, based on an analysis of UMS, is that the safety implications of user modifications to safety-critical software in a commercial nuclear power plant are a concern and the ability to modify safety-critical software without regulatory oversight or involvement should be prohibited. The additional effort to approve or certify UMS and to qualify the tools used to modify the software is significant, costly, and complex. Therefore, ISL considers that UMS is not appropriate for commercial nuclear power and might well be a security risk.

Section 9 of FAA Order 8110.49 clarifies the intent of RTCA DO-178B, Section 12.2, and its application with respect to tool qualification. Tool qualification requirements have changed from what is described in RTCA DO-178B; therefore, Section 9 of FAA Order 8110.49 no longer describes the state of the practice for tool qualification. New tool qualification requirements are discussed in RTCA DO-178C, Section 12.2. Also, the tool qualification criteria, objectives, and activities described in RTCA DO-178B have been expanded and moved to a supplemental document, RTCA DO-330. Even though RTCA DO-178B no longer represents the state of the practice, a discussion of an analysis of Section 9 of FAA Order 8110.49 is presented in the following paragraphs to clarify certain tool qualification issues that may be suitable for consideration as a technical basis for regulatory guidance for commercial nuclear power.

As stated in both RTCA DO-178B and RTCA DO-178C, qualification of a tool is only required if one of the software development processes is eliminated, reduced, or automated by the use of that tool without its output being verified. Section 9 of FAA Order 8110.49 clarifies this to mean that if the results of the tool are being relied on to supply the sole evidence that one or more objectives are satisfied, the tool must be qualified. However, if the result of the verification activity performed by the tool is confirmed by another verification activity, then there is no need to qualify the tool. An ISL observation, based on our analysis of this clarification, is that all tools used to eliminate, reduce, or automate software development processes which have not been verified must be qualified if there is any potential of introducing errors into the software product. However, for tools designed to detect errors in software and which cannot introduce errors into the software product, the tool does not need to be qualified if the output of the tool can be verified by using another verification activity. This statement illustrates the difference in the way the FAA deals with tools that can introduce errors compared to tools that may fail to detect errors. This philosophy still persists in RTCA DO-178C and RTCA DO-330 although the details of tool qualification have changed from RTCA DO-178B. ISL recommends that the NRC, in any new regulatory guidance, should consider implementing different tool qualification requirements depending on whether the tool can introduce an error or fail to detect an error consistent with the FAA and RTCA.

Consistent with RTCA DO-178B, Section 9 of FAA Order 8110.49 considers that only two types of tools need to be considered for qualification (i.e., software verification tools and software

development tools). Verification tools are defined by RTCA DO-178B as tools that cannot introduce errors but may fail to detect them. Development tools are defined by RTCA DO-178B as tools whose output is part of airborne software and thus can introduce errors. In RTCA DO-178C, the terms development and verification have been replaced with three tool criteria. Criterion 1 covers RTCA DO-178B development tools and Criteria 2 and 3 cover slightly different types of RTCA DO-178B verification tools. A simple analysis of the RTCA DO-178B definition of a development tool or Criterion 1 from RTCA DO-178C could lead to the conclusion that a development or Criterion 1 tool is limited to a tool that directly generates source code or libraries that are compiled and linked to generate the executable object code since this meets the rigid definition of being a part of airborne software. However, based on a more detailed analysis of these types of tools as supported by answers to frequently asked questions documented in RTCA DO-330, Appendix D, Section 1.5.3.3.1, ISL observes that a development or Criterion 1 tool includes any tool that automatically produces a part of the outputs of any one of the software development processes. This encompasses tools that transform high-level requirements into design requirements, design requirements into low-level requirements, and low-level requirements into source code. It also covers tools that generate data files, configuration files, or the executable object code that could inject an error into any of the outputs of the development process. ISL recommends that any incorporation of tool qualification guidance into the commercial nuclear power regulatory process should not rely on clarifications in other documents and should clearly define the type of tool that could introduce errors into the executable object code.

Section 13 of FAA Order 8110.49 discusses policies for properly overseeing suppliers. The policies apply when suppliers and sub-tier suppliers perform system and software development, verification, and certification activities. Although Section 13 of FAA Order 8110.49 does not specifically mention software tools, overseeing tool development by suppliers should follow the policies when possible. However, since COTS tools are a pre-developed piece of general software intended for widespread use by multiple users in any number of software development projects, overseeing the vendor's activities may not be feasible. ISL recommends that the NRC consider COTS tools in any regulatory guidance and how vendor oversight might be achieved.

Section 14 of FAA Order 8110.49 discusses policies for software problem reporting. The policies apply when an applicant's suppliers and sub-tier suppliers will be responsible for managing problems detected during the development of aircraft systems implemented with software. The applicant should discuss in their SCM plan, or other appropriate planning documents, how they will oversee their supplier's and sub-tier supplier's software problem reporting process. These plans should describe any tools that the applicant's suppliers or sub-tier suppliers plan to use for the purpose of recording action items or observations for the applicant to review and approve prior to entering them into the applicant's problem reporting system. ISL recommends that the NRC consider the need for adequate discussion and planning for tools used by suppliers in any new regulatory guidance to track software problem and reporting processes.

Section 16 of FAA Order 8110.49 discusses the policies associated with management of software development or verification environments. The policies apply when a software

development or verification environment is used that may not be completely representative of the target computer. Configuration control of the environment should be established and maintained and a structured problem reporting system for the environment should be implemented that is available to users of the environment. The software development plan and software verification plan should include an explanation of how the software development or verification environment will be used to show compliance with RTCA DO-178B objectives. With respect to software tools, the only relevant policy in this section is that if development tools are being used in the integrated environment, then verification should also be performed in the integrated environment. ISL recommends that the NRC consider, in any new regulatory guidance, implementing a similar policy of tool verification in the integrated environment.

2.2.2 Summary

FAA standards, advisory circulars, and orders along with RTCA documents provide comprehensive, well-organized software tool requirements, guidance, and review and approval practices that would form a sound technical basis for developing new regulatory guidance for the commercial nuclear power industry.

FAA-STD-026A is a high-level standard that primarily establishes and communicates requirements to be applied during the development and support of computer software. FAA-STD-026A provides minimal guidance on the use, verification, validation, qualification, review, or approval of tools used during software development. The standard simply requires the identification, description, and validation of tools but does not supply any guidance or define any processes for the validation of tools. The identification and description of software tools is addressed through specific documentation requirements that define the format and content of documents that discuss data generated during the software development process and detailed information on software tools used during the software development process.

RTCA DO-178C and RTCA DO-330 provide a comprehensive tool qualification process with RTCA DO-178C defining the need and rigor necessary for the tool qualification while RTCA DO-330 provides the processes, activities, and tasks for tool qualification. RTCA DO-178C recognizes five software levels, Level A through Level E, where Level A corresponds to a catastrophic loss of aircraft and Level E corresponds to no impact on aircraft, crew, or safety. The software level is determined based upon the failure condition that may result from anomalous behavior of the software as determined by a system safety assessment. Guidance for performing a system safety assessment is provided in FAA Order 8110.49. The software level of a tool is consistent with the software level of the software component developed with the tool. RTCA DO-178C discusses partitioning as a technique for providing isolation between software components to contain and isolate faults and potentially reduce the effort of the software verification process and tool qualification process. RTCA DO-178C stresses the importance of planning including choosing design methods, tools, and programming languages that limit the opportunity for introducing errors, and verification methods that ensure that errors introduced are detected. RTCA DO-178C allows the use of optimization when compiling source code provided that proper planning is performed to determine the impact of the optimization.

RTCA DO-178C determines if a tool needs to be qualified and the rigor of qualification. A tool needs to be qualified only if it reduces, eliminates, or automates a software development life cycle process, the output of the tool or the tool itself has not been verified, and the tool is used to develop software that has an impact on the aircraft operational capability, safety, or crew workload. Most, if not all, commercially developed software tools cannot be fully verified due to the proprietary nature of the tool development and the unavailability of verification data. Therefore, commercially developed tools must almost always be qualified. If a tool needs to be qualified, a tool qualification level of TQL1 to TQL5 is assigned to the tool to determine the amount of rigor necessary to qualify the tool. The tool qualification level depends on the level of software as determined by a system safety assessment and the type of tool. The determination of the software level and the type of tool are both deterministic processes that assess how potential failures could impact the aircraft and crew. Risk and probability are not used in the FAA tool qualification process.

RTCA DO-330 presents an entire qualification and development process for software tools that mirrors the software development processes described in RTCA DO-178C, tailored for software tool development. All of the tool qualification processes described in RTCA DO-330 apply to software tools being developed as part of a software project, either in-house or through a specific outside vendor or contractor, where the responsibility of tool qualification is shared by both the tool developer and the tool user. Only a limited number of the tool qualification processes apply to COTS tools because the tool has already been commercially developed and made available to the public for general use. COTS tools need to comply with the objectives appropriate to the necessary TQL assigned to the tool with tool qualification activities split between the tool developer and the tool user. A tool is considered qualified when all the tool development life cycle process objectives and activities based on the assigned TQL have been completed. RTCA DO-330 also provides guidance on tool qualification related to the use of multi-function tools, previously qualified tools, service history, exhaustive input testing, formal methods, and dissimilar tools for achieving compliance with objectives of RTCA DO-330.

An ISL observation, based on our analysis of RTCA DO-178C and RTCA DO-330, is that the guidance would form a sound technical basis for developing commercial nuclear power regulatory guidance; however, some changes to the guidance would be beneficial to provide a more comprehensive and practical process for software tool qualification as an approach to review and approval by the NRC and as a commercial-grade dedication approach to be used by users or third-party dedicators. One of the changes would involve either adding more tool qualification levels or redefining the tool qualification levels associated with the most safety-critical software, Levels A and B. The current guidance for TQL-5 tools (i.e., verification tools) does not require most tool qualification process objectives to be satisfied even though these verification tools are used to find defects in Level A and B software. A failure in Level A or B software could cause a catastrophic or hazardous impact on the aircraft or crew. Similarly, an analogous failure in safety-related and highly risk-significant software in a nuclear power plant digital instrumentation and control (DI&C) system could create a substantial safety hazard as defined in 10 CFR 21.3. The responsible flaw in the software that might have been introduced or allowed to remain undetected by a faulty tool would be equivalent to a defect as defined in 10 CFR 21.3, including the flaw's potentially creating a substantial safety hazard or leading to

exceeding of a technical specification safety limit. An ISL observation, based on our analysis, is that tools used to verify Level A and B software should be held to a higher standard than tools used to verify Level C and D software.

Another change to the RTCA DO-330 tool qualification requirements relates to COTS tools. The tool qualification process for COTS tools requires considerable participation by the COTS tool vendor which is problematic. Beyond a detailed user's manual, a COTS tool vendor would probably not provide the necessary qualification data required by these documents. Also, the COTS tool development process is proprietary, not subject to oversight or review, and probably does not meet the tool qualification process objectives specified in these documents. COTS tool qualification should follow a more practical approach, without being able to determine the adequacy of the vendor's proprietary process, by a combination of performance history review and analysis in conjunction with the use of one or more alternate methods of qualification such as exhaustive input checking or the use of dissimilar tools.

FAA AC 20-115C is a recent advisory circular that is similar to an NRC regulatory guide because it endorses the use of RTCA DO-178C, RTCA DO-330, and RTCA DO-330 supplements RTCA DO-331, RTCA DO-332, and RTCA DO-333 as a means of qualifying tools, providing guidance for FAA review of tool use, and obtaining FAA approval of tools. FAA AC 20-115C encourages the use of these endorsed documents for qualifying new tools or re-qualifying modified tools. It also addresses whether previously qualified tools need to be re-qualified or whether they can be used without re-qualification. FAA AC 20-115C also describes the required objectives, activities, and data that must be submitted to facilitate review and approval of the tool.

FAA Order 8110.49 provides guidance and clarifies the software and software tool review and approval processes described in RTCA DO-178B. Although FAA Order 8110.49 does not currently recognize the processes of RTCA DO-178C and RTCA DO-330, most of the guidance and clarifications of FAA Order 8110.49 apply to the updated processes in those documents. FAA Order 8110.49 describes four types of reviews performed by the certification authority. The four reviews include a software planning review, software development review, software verification review, and final certification review. Tool qualification plans and tool qualification data need to be provided for these reviews. FAA Order 8110.49 also defines the level of FAA involvement in software projects. For software level A or B projects, the use of unusual tools requires a HIGH level of FAA involvement. For software level C or D projects, the use of unusual tools requires at least a MEDIUM level of FAA involvement. An ISL observation, based on our analysis of FAA involvement, is that tool complexity, tool familiarity, tool supplier, and tool service history should all be factors to consider when determining the level of involvement rather than the tool just being unusual.

The FAA standards, advisory circulars, and orders identify and recommend tools for many of the software engineering processes. RTCA documents provide a comprehensive tool development and qualification process. Based on the analysis of FAA and RTCA documents, the following essential elements of the documents relative to software tools have been determined.

- (1) All software not developed under contract, including tools and COTS software, is regarded as nondevelopmental software.
- (2) All software tools need to be identified and addressed within required software development activities and documentation.
- (3) All tools need to be validated to ensure that the tool performs its required functions and operates correctly within the software system design.
- (4) Tools only need to be qualified if the tool has not been verified and the tool eliminates, reduces, or automates a software development life cycle process.
- (5) Software is categorized as Level A, B, C, D, or E depending on how an error in a software component relates to the system failure condition and the severity of the failure condition.
- (6) The software level of a tool is consistent with the software level of the software component created with the tool based upon the failure condition that may result from anomalous behavior of the software as determined by the system safety assessment.
- (7) Choose design methods, tools, and programming languages that limit the opportunity for introducing errors, and verification methods that ensure that errors introduced are detected.
- (8) Using optimization when compiling source code is acceptable provided that proper planning is performed to determine the impact of the optimization.
- (9) The amount of rigor necessary to qualify a tool is determined through the assignment of a tool qualification level (TQL) which depends on the type of tool and the safety level of the software which is being developed using the tool.
- (10) RTCA DO-178C recognizes three basic types of tools to determine tool qualification requirements: developmental tools, and two types of verification tools that are distinguished by the number of life cycle processes reduced, eliminated, or automated.
- (11) A TQL of 1 to 5 is assigned to determine the amount of rigor necessary to qualify the tool.
- (12) Qualification activities can be modified from the activities specified in RTCA DO-178C by providing evidence of the dissimilar nature of multiple-version dissimilar software and dissimilar simulators.
- (13) Frequent communication with the approval authority as part of the tool qualification liaison process is paramount for obtaining approval of tools.

- (14) Qualification of COTS tools requires a cooperative effort between the tool developer and the tool user.
- (15) Review and approval of software tools is dependent on the applicant following all the objectives and developing all the associated life cycle data as required by RTCA DO-178C, RTCA DO-330, and all its supplements.
- (16) FAA Order 8110.49 describes four reviews that are integral to the review and approval process of software and software tools including a software planning review, a software development review, a software verification review, and a final certification software review.
- (17) The level of FAA involvement in software projects is LOW, MEDIUM, or HIGH depending on the software level, software certification experience, software development capability, software service history, software complexity, and experience and capabilities of the FAA certification authority's designated engineering representative or person responsible for software oversight.
- (18) The use of unusual tools increases the level of FAA involvement. For software level A or B projects, the use of unusual tools requires a HIGH level of FAA involvement. For software level C or D projects, the use of unusual tools requires at least a MEDIUM level of FAA involvement.

Based on the analysis of FAA and RTCA documents, ISL has made the following important observations and recommendations.

- (1) Partitioning, multiple-version dissimilar software, and safety monitoring may change the software level of the software component and the assigned level of the tool used to develop the software component and reduce the amount of rigor necessary to qualify the tool and verify the software.
- (2) If a tool is a commercially available product, source code review and analysis, verification of low-level requirements used to create the source code, verification of the software architecture, and verification of the degree of test coverage may not be possible due to the proprietary nature of the tool development; therefore, the tool must be qualified.
- (3) The NRC should consider regulatory guidance that adopts similar processes to RTCA DO-178C and RTCA DO-330 that would determine tool qualification requirements based on a two-dimensional, deterministic process of assigning a software level based on a safety assessment of the system containing the software and performing a tool impact assessment to determine the type of tool and the tool's impact on the software development life cycle processes.
- (4) The rigor of TQL-5 tool qualification should be increased because verification tools classified as TQL-5 can fail to detect Level A or B software defects that can potentially cause a catastrophic or hazardous failure of the aircraft.

- (5) The categorization of some of the RTCA DO-330 tool development objectives as discretionary should be reconsidered for TQL-4 and TQL-5 tools.
- (6) The NRC should consider regulatory guidance that adopts the RTCA DO-330 tool qualification liaison process as part of the NRC software tool review and approval process.
- (7) A more practical approach to COTS tool qualification is required that does not rely on vendor participation and generation of qualification data beyond what is in the user's manual. COTS tool qualification should concentrate on the use of service history, exhaustive input testing, formal verification methods, or dissimilar tools.
- (8) FAA Advisory Circular 20-115C is a sound technical basis for developing a software tool review and approval process.
- (9) The NRC should consider regulatory guidance that stresses the need for adequate planning.
- (10) The level of FAA involvement in software projects should be higher for tools other than those classified as unusual. Tool complexity, tool familiarity, tool supplier, and tool service history should all be factors to consider when determining the scope of the review and approval process.
- (11) The safety and security implications of user modifications to safety-critical software in a commercial nuclear power plant are a concern and the ability to modify safety-critical software without regulatory oversight or involvement should be prohibited.
- (12) The NRC should consider regulatory guidance that covers verification reviews of both developed and COTS tools.
- (13) The NRC should consider regulatory guidance that implements different tool qualification requirements depending on whether the tool can introduce an error or fail to detect an error.

2.3 Nuclear Regulatory Commission Analysis

The Atomic Energy Commission (AEC) was established by the Atomic Energy Act of 1946 to regulate nuclear power. Commercial nuclear power became possible through the Atomic Energy Act of 1954, which encouraged the use of civilian nuclear power and regulated its safety. The AEC sought to ensure public health and safety from hazards of nuclear power without imposing excessive requirements that would inhibit the growth of industry. The Nuclear Regulatory Commission (NRC) was established by the Energy Reorganization Act of 1974 and continued the regulatory mission of the AEC, focusing on broad issues that were essential to protecting public health and safety, but independent of the developmental and promotional aspects of the former AEC's mission. The NRC formulates policies, develops regulations

governing nuclear reactor and nuclear material safety, issues regulatory guidance and generic communications, licenses various types of nuclear facilities and related activities, licenses facility operators, licenses the industrial and medical use of special nuclear materials, provides regulatory oversight of its licensed and regulated facilities and activities, and takes enforcement actions for violations of regulations, including orders to licensees, revocation of licenses or permits, civil penalties, and even criminal penalties. The NRC's regulatory activities are focused on safety oversight of all its licensed facilities and activities, reactor license renewal of existing nuclear plants, and evaluation of new applications for nuclear plants and other nuclear facilities and activities. The NRC's regulatory process has five main components, one of which is developing regulations and guidance for applicants and licensees. The NRC develops and amends regulations that licensees must meet to obtain or retain a license to operate a nuclear facility and develops and revises regulatory guides and other guidance to aid licensees in meeting safety requirements. It also develops review guidance, such as standard review plans, to prescribe the process and acceptance criteria for NRC staff review of various types of applications for approval or endorsement of various types of nuclear facilities, equipment, software, programs, and processes. The NRC also works with industry organizations, including standards organizations, to develop consensus standards and other publications so that NRC guidance can reference and endorse these standards. NRC endorsement of a standard means that the NRC staff determines that meeting the standard, with certain NRC-specified qualifications, is one approved means of compliance with one or more regulatory requirements (i.e., legal requirements of the regulations, licenses, or orders of the NRC).

2.3.1 Document Analysis

Several regulatory guides (RGs), standard review plan (SRP) branch technical positions (BTPs) and appendices, interim staff guidance (ISG), nuclear regulatory publications (NUREGs) and contractor reports (NUREG/CRs) discuss software tools. The following paragraphs discuss the analysis of relevant NRC documents including RG 1.152, RG 1.168 through 1.173, ISG-06, NUREG 0800 BTP 7-14 and Appendix 7.1-D, NUREG/CR-6263, and NUREG/CR-6421.

In addition, there are other NRC regulations and regulatory guidance documents that would ultimately be applicable to software tools and their use, but at a high level in that they govern the safety classification of all nuclear facility structures, systems, components, equipment, software, and related services (i.e., whether they are important to safety, including those that are safety-related and certain ones that are nonsafety-related), provide QA requirements, provide requirements for reporting of defects and noncompliance, and provide requirements for commercial-grade dedication. However, while citing these regulations as appropriate, including 10 CFR Part 50, Appendix B, and 10 CFR Part 21, ISL will defer detailed discussion of these higher-level documents to Task 3 in which ISL will provide the technical and regulatory bases for possible guidance, and ways in which such guidance could fit into the NRC regulatory framework.

2.3.1.1 Regulatory Guide 1.152

RG 1.152, Revision 3, describes a method that the NRC considers acceptable to comply with several NRC regulations for promoting high functional reliability, design quality, and a secure

development and operational environment for the use of digital computers in the safety systems of nuclear power plants. The applicable regulations include 10 CFR Part 50, 10 CFR 50.55a(h), General Design Criterion 21 of Appendix A to 10 CFR Part 50, and Criterion III, "Design Control," of Appendix B to 10 CFR Part 50. The analysis of this latest revision of RG 1.152 is discussed in the following paragraphs.

RG 1.152 discusses several documents (i.e., IEEE 7-4.3.2-2003, 10 CFR 50.55a(h), IEEE 279-1971, and IEEE 603-1991) that address barriers between safety and nonsafety functions in digital computers used in nuclear power plants. IEEE 7-4.3.2-2003 does not require nonsafety software to meet the requirements of IEEE 7-4.3.2-2003 if barriers between safety and nonsafety software are designed in accordance with the requirements of IEEE 7-4.3.2-2003 and do not interfere with the performance of the safety functions of the safety software or firmware. However, 10 CFR 50.55a(h) requires that nuclear power plants conform either to IEEE 279-1971 or IEEE 603-1991 which require equipment that performs both safety and nonsafety functions to be classified as part of the safety system. Therefore, in its qualified endorsement of IEEE 7-4.3.2-2003, RG 1.152 takes exception to the IEEE 7-4.3.2-2003 position on nonsafety software and clarifies that any software providing nonsafety functions that resides on a computer providing a safety function must be classified as a part of the safety system. An ISL observation, based on our analysis of the barriers between safety and nonsafety software is that the barrier requirements of IEEE 7-4.3.2-2003 are not relevant to software tools and are only applicable to system hardware and system software. ISL further observes that requiring equipment that performs both safety and nonsafety functions on the same computer to be classified as part of the safety system appears inconsistent with the classification and commercial-grade dedication of safety-related and nonsafety-related software based on their importance to safety as described in 10 CFR Part 50, Appendix B, and 10 CFR Part 21.

An ISL observation, based on an analysis of software classification requirements, is that while IEEE-603-1991 (invoked by 10 CFR 50.55a(h)), calls for classification of software in a safety system, but with nonsafety functions as part of the safety system, IEEE 603-2009 does not. Instead, IEEE 603-2009 states that equipment performing nonsafety functions that is connected to safety-related equipment should either be classified as an associated circuit or shall be electrically isolated from the safety system, have digital communication independence, and be classified as non-Class 1E. The RG 1.152 position on the classification of nonsafety functions also negates the benefits of software partitioning which can isolate the nonsafety functions from the safety functions to minimize the cost and manpower required to qualify the nonsafety portions of the system because nonsafety systems do not require commercial-grade dedication for use in commercial nuclear power plants. An ISL observation, based on our analysis, is that the NRC should consider updating RG 1.152 to address the IEEE 603-2009 changes to the classification of nonsafety-related systems, and should address an approach to software classification and dedication based on the importance of the software to the safety of the system.

Except for the exception noted above, RG 1.152 states that conformance with the requirements of IEEE 7-4.3.2-2003 is a method that the NRC staff has deemed acceptable for satisfying the NRC's regulations with respect to high functional reliability and design requirements for

computers used in safety systems of nuclear power plants. However, the NRC has not endorsed Annexes B through F of IEEE 7-4.3.2-2003 because the NRC considers that the guidance is insufficient or inadequate. An ISL observation, based on an analysis of the guidance that conformance with the requirements of IEEE 7-4.3.2-2003 is an acceptable method for satisfying NRC regulations, is that this guidance applies to software tools since Section 5.3.2 of IEEE 7-4.3.2-2003 specifies methods that must be used to confirm that software tools are suitable for use. However, IEEE 7-4.3.2-2003 has been superseded by IEEE 7-4.3.2-2010 and IEEE 7-4.3.2-2010 will be superseded by a new version of IEEE 7-4.3.2 that is currently in development. The NRC should consider updating RG 1.152 to recognize the latest version of IEEE 7-4.3.2.

The main focus of the regulatory positions in RG 1.152 is to provide specific guidance for establishing and maintaining a secure development and operational environment for the protection of digital safety systems throughout the design and development life cycle phases of the safety system. RG 1.152 promulgates regulatory positions for five phases of system development including the concept phase, requirements phase, design phase, implementation phase, and testing phase.

Regulatory Position 2.1: Regulatory guidance applicable to the concept phase of system development states that design features to establish and maintain a secure operational environment for the system should be identified and described as part of the license application. Further, the susceptibility to inadvertent access and undesirable behavior from connected systems should be analyzed and the analysis should be used to establish design feature requirements for both hardware and software to establish and maintain a secure operational environment. Finally, regulatory guidance states that remote access to the safety system should not be allowed (i.e., ability to access a computer, node, or network resource that performs a safety function from a computer or node located in an area with less physical security than the safety system).

Regulatory Position 2.2: Regulatory guidance applicable to the requirements phase of system development states that the functional performance requirements and system configuration for a secure operational environment should be defined and should be part of the overall system requirements. Software tools are not mentioned in this regulatory position; however, the position does mention pre-developed software (i.e., reused software and COTS software). The regulatory guidance states that requirements specifying the use of pre-developed software and systems should address the reliability of the safety system by taking into consideration previous testing of pre-developed software functions and functions that are supported by operating experience.

Regulatory Position 2.3: Regulatory guidance applicable to the design phase of system development states that the safety system design features identified in the requirements phase for a secure operational environment should be translated into specific design items in the design phase. Safety system design items should

address control of physical and logical access to system functions, use of safety system services, and data communication with other systems. Also, the design phase should address how pre-developed software used in the safety system may challenge the secure operational environment for the safety system. Regulatory guidance also states that measures should be taken to prevent the introduction of unnecessary design features that may result in unwanted or unnecessary code.

Regulatory Position 2.4: Regulatory guidance applicable to the implementation phase of system development states that the developer should ensure that a secure operational environment is implemented correctly, accurately, and completely based on the design requirements. The developer should also implement a secure development environment to preclude alterations to the developed system. Regulatory guidance also states that the developer should ensure that COTS software or system features do not compromise the required design features of the secure operational environment.

Regulatory Position 2.5: Regulatory guidance applicable to the test phase of system development states that the design features of the secure operational environment should be tested to ensure system reliability. Each system design feature of the secure operational environment should be validated to verify that the implemented feature achieves its intended function.

An ISL observation, based on our analysis of these regulatory positions, is that establishing and maintaining a secure operational environment and a secure development environment for system software is not relevant to software tools. Neither the operational environment nor development environment of the software tool needs to be secure even if the software tool is used to develop safety-related software because tools are not likely to be hacked while being developed and a would-be attacker of the application software would not know which tool was used or would be used in its development.

2.3.1.2 *Regulatory Guide 1.168*

RG 1.168, Revision 2, describes a method acceptable to the NRC staff for satisfying the NRC's regulations with respect to software V&V and with respect to conducting audits, inspections, walkthroughs, and reviews. The analysis of this latest revision of RG 1.168 is discussed in the following paragraphs.

For safety system software, software V&V, reviews, inspections, walkthroughs, and audits are methods used to achieve compliance with the regulatory requirements of Appendices A and B of 10 CFR Part 50. RG 1.168 endorses IEEE 1012-2004 which provides guidance acceptable to the NRC staff for V&V of safety system software. An ISL observation, based on our analysis of this endorsement, is that the NRC should consider endorsing IEEE 1012-2012. IEEE 1012-2012 rearranges the software V&V activities to facilitate understanding and ease of use. IEEE 1012-2012 also adds system and hardware V&V activities to the existing software V&V activities and modifies the terminology to be consistent with the latest standards devoted to system, hardware, and software life cycle processes, IEEE 12207-2008 and IEEE 15288-2008. Life

cycle processes are relevant to software tools because the requirements and guidance for the selection, use, verification, validation, qualification, review, and approval of tools vary within each of the different life cycle processes. A thorough discussion of IEEE 12207-2008 and IEEE 15288-2008 is contained in Section 2 of ISL-ESD-TR-13-05 which is the Task 1 report from this research project.

RG 1.168 also endorses IEEE 1028-2008 which provides guidance acceptable to the NRC staff for carrying out software reviews, inspections, walkthroughs, and audits. IEEE 1028-2008 also contains an anomaly ranking and reporting scheme that should be considered by applicants and licensees when addressing the anomaly reporting documentation requirements of IEEE 829-2008. Meeting these standards helps to meet regulatory requirements by ensuring that disciplined software V&V, review, and audit practices are incorporated into software processes applied to safety system software.

RG 1.168 describes software criticality through its endorsement of IEEE 1012-2004. IEEE 1012-2004 defines the method of quantifying software criticality using a software integrity level (SIL) where SIL 4 is the highest category and SIL 1 is the lowest. The SIL determines the minimum V&V activities that are required. RG 1.168 takes exception to the graduated SIL classifications of IEEE 1012-2004 and states that software used in nuclear power plant safety systems should be assigned a SIL 4 classification. An ISL observation, based on an analysis of this regulatory position, is that classifying all safety-related software as SIL 4 does not allow the V&V activities to be tailored based on the importance to safety of the various elements of the safety-related system as specified in 10 CFR Part 50, Appendix B, Criterion II, "Quality Assurance Program."

RG 1.168 discusses V&V of COTS software intended for implementation in a safety-related system. The regulatory position emphasizes that when an external organization performs V&V, the extent of independence between the organization responsible for design and the organization responsible for verification and checking of the design must meet the independence requirements of 10 CFR Part 50, Appendix B. Also, the guidance in IEEE 1012-2004 for V&V of pre-existing software is not endorsed. RG 1.168 states that RG 1.152 and EPRI TR-106439 provide information on the acceptance of pre-existing software that is acceptable to the NRC.

The discussion of software tools in RG 1.168 is limited to a high-level discussion of how tools should be handled and when software tool V&V activities are required. RG 1.168 states that tools used in the development of safety system software should meet the requirements of IEEE 7-4.3.2-2003. RG 1.168 also states that the provisions of the regulatory guide only apply to software tools when V&V activities cannot detect flaws in the software produced by the tools. If V&V activities can detect flaws introduced into the software produced by the tools, then the V&V tasks of witnessing, reviewing, and testing of the software tools are not required. An ISL observation, based on our analysis of this regulatory position, is that the position only applies to tools that are used in software coding processes that have the capability of introducing defects into the safety system software. The regulatory position does not cover V&V tools, requirements management tools, CM tools, and QA tools since these types of tools cannot

introduce defects in safety system software. ISL recommends that the NRC consider expanding the guidance of RG 1.168 to cover all types of tools used in the software development process.

2.3.1.3 *Regulatory Guide 1.169*

RG 1.169, Revision 1, discusses CM plans for digital computer software used in safety systems of nuclear power plants. RG 1.169, Revision 1, endorses IEEE 828-2005, with some exceptions, whose methods are deemed acceptable to the NRC staff for planning and executing a SCM program. IEEE 828-2005 applies to general-purpose software development and maintenance efforts and does not have specific criteria for safety system software. ISL's analysis of this latest revision of RG 1.169 is discussed in the following paragraphs.

RG 1.169 discusses SCM with some applicability to software tools. RG 1.169 states that for safety system software, support software (i.e., tools) used in the development of safety system software and testing tools used to test safety system software must be identified and controlled as configuration items or discussed in controlled documents. RG 1.169 also states that items that may not change but are necessary to ensure correct software production, such as compilers, should also be configuration items, thereby ensuring that all factors contributing to the executable software are understood. An ISL observation, based on our analysis of SCM regulatory guidance, is that RG 1.169 could be interpreted to cover all types of tools; however, the guidance could also be interpreted to only apply to testing, design, and implementation tools. The confusion exists because of the use of the phrase "used in the development of safety system software." Since testing is part of the life cycle of software development, identifying testing tools along with support tools used in software development implies a more restrictive definition of development. ISL recommends that the NRC consider modifying RG 1.169 to more clearly indicate that all tools are covered by the guidance instead of relying on an interpretation of the meaning.

RG 1.169 discusses control of purchased materials including COTS software. RG 1.169 states that contractually developed or qualified commercial software products that are safety system software must be taken under control by an SCM program. The exact version of the product must be identified and controlled according to the change control procedures applied to other configuration items and that its usage is tracked and reported.

RG 1.169 discusses development tools which, as discussed above, could mean all support tools or just design and implementation tools. RG 1.169 states that tools used in the development of safety system software should be handled according to IEEE 7-4.3.2-2003 and must be taken under control (i.e., treated as a configuration item) by an SCM program operated by the using organization that complies with IEEE 828-2005.

2.3.1.4 *Regulatory Guide 1.170*

RG 1.170, Revision 1, discusses software and system test documentation for digital computer software used in safety systems of nuclear power plants. RG 1.170, Revision 1, endorses IEEE 829-2008 as a means to comply with NRC regulations with respect to the documentation of software testing activities. IEEE 829-2008 defines requirements for software test

documentation including the form and content of the documentation. The analysis of this latest revision of RG 1.170 is discussed in the following paragraphs.

Documentation requirements in IEEE 829-2008 cover test planning, test specifications, and test reporting. Test planning requires test plans that address the scope, risks, tasks, resources, responsibilities, and acceptance criteria for the software item being tested. The test plan uses an integrity scheme level that includes a life cycle phase and a traceability matrix for software projects. Test specifications consist of test designs, test cases, and test procedures for testing the software item. The test procedures contain the detailed procedures and instructions for testing and the acceptance criteria for the test. Test reporting includes an interim status report, an anomaly report, test incident reports, test logs, and test summary reports. The final test summary report records and summarizes the test events. The final test summary report also follows the integrity scheme needed within the life cycle and forms the basis of evaluating test results.

RG 1.170 defines the minimum information necessary in software test documentation and identifies that test records must be maintained as quality records controlled by SCM. RG 1.170 also notes that each feature in safety system software should be formally tested under at least one test design. RG 1.170 also requires traceability in the test case documentation to ensure that all functions are tested. RG 1.170 states that the risk assessment scheme in IEEE 829-2008 and the use of four integrity levels for safety system software is unacceptable because of the potential catastrophic consequences of safety system software failure. Consistent with RG 1.168, RG 1.170 states that SIL 4 should be assigned to all safety system software.

Neither RG 1.168 nor RG 1.170 acknowledge 10 CFR 50.69 which provides a risk-informed categorization and treatment of structures, systems, and components (SSCs) for nuclear power reactors. As defined in 10 CFR 50.69, SSCs can be categorized using four risk-informed safety classes (RISC) (i.e., RISC-1 through RISC-4) based on the safety significance of the functions performed by the SSCs. RISC Classes 1 and 3 apply to safety-related SSCs that perform safety significant and low safety-significant functions, respectively. RISC Classes 2 and 4 apply to nonsafety-related SSCs that perform safety significant and low safety-significant functions, respectively. SSCs must be categorized as RISC-1, RISC-2, RISC-3, or RISC-4 by considering results and insights from the plant-specific probabilistic risk analysis (PRA) and by determining SSC functional important using an integrated, systematic process for addressing initiating events, SSCs, and plant operating modes, including those not modeled in the plant-specific PRA. Compliance with the requirements of 10 CFR 50.69 is voluntary and is, for RISC-3 and RISC-4 SSCs, an alternative to compliance with several regulations

The only statement in RG 1.170 relevant to software tools is an objection to the IEEE 829-2008 documentation requirements of information managed by automated testing tools. RG 1.170 takes exception to Clause 6.3 of IEEE 829-2008, which states that there is no need for repetition of test information in test documentation if the test information is completely managed by an automated tool, provided that the test documentation contains references for tracing the information. The NRC points out that repetition is sometimes needed and beneficial to make test information easily accessible to the NRC staff to improve the timeliness of a safety

conclusion. RG 1.170 states that test information should be available within the record management system even if that documentation of the test information is repetitive with documentation generated by the automated testing tools.

An ISL observation, based on our analysis of RG 1.170, is that the guidance for software tools is limited. RG 1.170 only identifies that tools used should be documented and test information should be available within the record management system even if it is repetitious with documentation generated by automated testing tools. However, this limited guidance on documentation of tools used is acceptable because detailed information on the tool (e.g., acquisition, training, support, and qualification) is required to be documented by IEEE 829-2008.

With respect to software testing life cycle activities, RG 1.170 refers to IEEE 1074-2006 for current practice using software life cycle processes that incorporate software testing activities. An ISL observation, based on our analysis of software life cycle processes, is that the NRC should consider revising RG 1.170 to reference IEEE 12207-2008 and IEEE 15288-2008 instead of IEEE 1074-2006 for system and software testing life cycle processes and activities because IEEE 12207-2008 is an internationally accepted standard of software life cycle processes and IEEE 15288-2008 is an internationally accepted standard of system life cycle processes while IEEE 1074-2006 is simply an IEEE standard that can be used to develop life cycle processes for specific applications such as the processes that are already specified in IEEE 12207-2008 and IEEE 15288-2008. A thorough discussion of the similarities and differences between IEEE 1074-2006, IEEE 12207-2008, and IEEE 15288-2008 is contained in Section 2 of ISL-ESD-TR-13-05 which is the Task 1 report from this research project.

Another ISL observation, based on our analysis of RG 1.170, is that the requirement to assign a classification of SIL 4 to nuclear safety system software is consistent with RG 1.168. However, based on our analysis of this regulatory position, ISL observes that the RISC-based categorization of SSCs is not recognized by RG 1.168 or RG 1.170 and that classifying all safety-related software as SIL 4 does not allow tailoring of the content of test documentation to the importance of the safety-related software for consistency with 10 CFR Part 50, Appendix B, Criterion II. Using multiple SILs for classifying nuclear plant safety-related software would allow the requirements for testing documentation to be applied commensurate with the software's importance to safety to be consistent with 10 CFR Part 50, Appendix B, Criterion II. A recognized method of selectively applying software development requirements commensurate with the software's importance to safety is discussed in the EPRI TR-102260 commercial-grade dedication process.

2.3.1.5 *Regulatory Guide 1.171*

RG 1.171, Revision 1, discusses software unit testing for digital computer software used in safety systems of nuclear power plants. RG 1.171, Revision 1, endorses IEEE 1008-1987 (reaffirmed in 1993 and 2002) with exception (explained below) as it applies to software unit testing. IEEE 1008-1987 defines a method for planning, preparing for, conducting, and evaluating software unit testing. Current practice for the development of software for safety-related applications uses a software life cycle process that incorporates software testing activities such as those specified in IEEE 1074-1991 and IEEE 1074-2006. Software testing,

including software unit testing, is a key element in software V&V activities. A common approach to software testing, as discussed in NUREG/CR-6101 and NUREG/CR-6263, utilizes a three-level test program including unit-level testing, integration-level testing, and system-level testing to help ensure quality in complex software products. ISL's analysis of this latest revision of RG 1.171 is discussed in the following paragraphs.

RG 1.171 states that the two aspects of unit test coverage that are particularly important for safety system software are coverage of requirements and coverage of the internal, modular structure of the code. With respect to requirements coverage, RG 1.171 states that unit testing should include all features and associated procedures, states, state transitions, and associated data characteristics essential to the safety determination instead of consideration of expected use of the safety system software as specified in IEEE 1008-1987. With respect to statement coverage, RG 1.171 does not endorse the use of statement coverage as a criterion for measuring the completeness of software unit testing and states that unit test coverage criteria should be identified and justified.

The only guidance in RG 1.171 relevant to software tools is a statement regarding the minimum documentation necessary to meet regulatory requirements. IEEE 1008-1987 mandates the use of the test design specifications and the test summary report documents defined in IEEE 829-2008 with a requirement for additional information to cover unit testing documentation. RG 1.171 states that the documentation to support software unit testing should include, as a minimum, special conditions and controls, equipment, tools, and instrumentation needed for the accomplishment of testing.

An ISL observation, based on our analysis of RG 1.171, is that the guidance for software tools is limited. RG 1.171 only identifies that tools used for software unit testing should be documented. This limited guidance on documentation of tools used is acceptable because detailed information on the tool (e.g., acquisition, training, support, and qualification) is required to be documented by IEEE 829-2008. RG 1.171 also refers to IEEE 1074-2006 for current practice using software life cycle processes that incorporate software unit testing activities. ISL recommends that the NRC consider revising RG 1.171 to reference IEEE 12207-2008 and IEEE 15288-2008 instead of IEEE 1074-2006 for system and software unit testing life cycle processes and activities. A thorough discussion of the similarities and differences between IEEE 1074-2006, IEEE 12207-2008, and IEEE 15288-2008 is contained in Section 2 of ISL-ESD-TR-13-05 which is the Task 1 report from this research project.

2.3.1.6 *Regulatory Guide 1.172*

RG 1.172, Revision 1, discusses a software requirements specification for digital computer software and complex electronics used in safety systems of nuclear power plants. In 10 CFR 50.55a(h), the NRC requires that reactor protection systems satisfy the criteria of IEEE 279-1971 which states that quality of components is achieved through the specification of design, inspection, and test requirements known to promote high quality. Several general design criteria of 10 CFR Part 50, Appendix A, also describe functions that would be included in a digital computer software requirements specification. ISL's analysis of this latest revision of RG 1.172 is discussed in the following paragraphs.

RG 1.172 endorses IEEE 830-1998 with exception as a method of complying with the general design criteria in Appendix A to 10 CFR Part 50 and the criteria for QA programs in Appendix B to 10 CFR Part 50 as they apply to the development of a software requirements specification. The software requirements specification is an essential part of the record of the design of safety system software. It serves as the design basis for the software to be developed, and is a crucial design input to the remainder of the software development process. The endorsed IEEE 830-1998 describes writing a software requirements specification for a wide variety of systems. While IEEE 830-1998 is not specifically aimed at safety applications, the standard does provide guidance on the development of a software requirements specification that will exhibit characteristics important for developing safety system software. Correct, complete, well-written, and unambiguous software requirements are essential for completing the design and verification processes of high-integrity software products.

The only software tool relevant guidance in RG 1.172 is concerned with requirements generated by specification or representation tools. IEEE 830-1998 discusses the use of specification or representation tools to generate requirements. Methods of generating requirements and the tools that support those methods can be object-oriented, process-based, or behavioral. The degree to which tools are useful in preparing the software requirements specification depends upon the size and complexity of the software. IEEE 830-1998 does not endorse any particular tool but states that it is best to retain the natural language descriptions for better understanding of the software requirements specification. RG 1.172 reinforces the IEEE 830-1998 requirements by stating that traceability should be maintained between representations generated by tools and the natural language descriptions of the software requirements that are derived from systems requirements and system safety analyses.

An ISL observation, based on our analysis of RG 1.172, is that it contains no specific guidance for software tools. ISL recommends that the NRC consider modifying RG 1.172 to require the documentation of detailed information on any tool used to develop a software requirements specification (e.g., acquisition, training, support, and qualification) or the NRC should attempt to influence the IEEE to update IEEE 830-1998 because tool information is not required to be documented by IEEE 830-1998. The NRC should also consider updating the endorsement of IEEE 830-1998 to IEEE 29148-2011. IEEE 830-1998 has been superseded by IEEE 29148-2011 and IEEE 29148-2011 specifies the required processes that must be implemented for the engineering of requirements for systems and software products throughout the life cycle. The requirements in IEEE 29148-2011 are specifically written for compatibility with the system and software life cycle processes defined in IEEE 15288-2008 and IEEE 12207-2008.

RG 1.172 also refers to IEEE 1074-2006 for a discussion of software life cycle processes that incorporates a software requirements specification as an essential input at the beginning of a software development life cycle. ISL recommends that the NRC consider modifying RG 1.172 to reference IEEE 12207-2008 and IEEE 15288-2008 instead of IEEE 1074-2006 for system and software life cycle processes and activities. A thorough discussion of the similarities and differences between IEEE 1074-2006, IEEE 12207-2008, and IEEE 15288-2008 is contained in Section 2 of ISL-ESD-TR-13-05 which is the Task 1 report from this research project.

2.3.1.7 Regulatory Guide 1.173

RG 1.173, Revision 1, discusses the development of software life cycle processes for digital computer software used in safety systems of nuclear power plants. RG 1.173, Revision 1, endorses IEEE 1074-2006 with exception as a method of designing a set of software life cycle processes and activities appropriate for the development of safety system software. IEEE 1074-2006 describes a set of processes and constituent activities that are commonly accepted as composing a controlled and well-coordinated software development process. IEEE 1074-2006 can be used as a basis for developing specific software life cycle processes that are consistent with regulatory requirements, as applied to software, for controlling and coordinating the design of safety system software. RG 1.173, Revision 1, states that IEEE 1074-2006 describes a complete set of software life cycle processes; however, it lacks a sufficiently detailed system-level perspective. Therefore, system-level aspects need to be addressed by other regulatory guidance and industry standards. ISL's analysis of this latest revision of RG 1.173 is discussed in the following paragraphs.

RG 1.173 does not discuss software tools but does address pre-existing software intended for use in a nuclear safety system application. RG 1.173 states that if pre-existing software (i.e., reusable software or COTS software) is to be incorporated into a safety system, then an acceptance process must be added at an appropriate point in the life cycle model to establish the suitability of the pre-existing software for its intended use.

An ISL observation, based on our analysis of RG 1.173, is that the regulatory guide contains no specific guidance for software tools and its relevance only lies with establishing a suitable and well-controlled set of safety system software development life cycle processes, activities, and tasks. IEEE 1074-2006 is a standard for developing software life cycle processes, and each organization that uses this standard must map the activities specified in the standard into its own software life cycle. However, IEEE 1074-2006 does not address system-level issues. Therefore, the overall system-level development processes must be developed in accordance with other regulatory guidance or industry standards. ISL recommends that the NRC consider revising RG 1.173 to endorse IEEE 12207-2008 and IEEE 15288-2008, which contain software and system development processes, respectively, and have IEEE, ISO, and IEC endorsement. Completing the required actions of IEEE 1074-2006 will produce a product (i.e., a software development process) similar to that already defined by IEEE 12207-2008; therefore, starting with IEEE 12207-2008 would save the time and expense of developing a suitable set of safety system software development life cycle processes. A thorough discussion of the similarities and differences between IEEE 1074-2006, IEEE 12207-2008, and IEEE 15288-2008 is contained in Section 2 of ISL-ESD-TR-13-05 which is the Task 1 report from this research project.

2.3.1.8 Interim Staff Guidance (ISG) 06

ISG-06 provides guidance for the NRC review of license amendments supporting installation of DI&C equipment in operating nuclear power plants originally licensed under 10 CFR Part 50. ISG-06 also identifies the documentation that should be reviewed and when the documentation should be provided. The documentation should allow technical reviewers to quickly assess DI&C upgrade applications to ensure that the reviewers sufficiently address the areas that are

within the I&C scope of review. ISG-06 references other guidance documents and standards, as appropriate, and identifies the context of the other guidance documents and standards with respect to the DI&C license amendment process for operating reactors.

ISG-06 recognizes two different methods of approving a component for use in safety-related applications based on whether the safety-related component critical characteristics can or cannot be verified. The first method is to dedicate a commercial-grade item (CGI). This is considered, if done correctly, to be equivalent to the second method, which is to design and manufacture the component under a 10 CFR Part 50, Appendix B-compliant QA program. If a safety-related component has critical characteristics that can be verified, then the component may be designed and manufactured as a CGI and commercially dedicated. If the critical characteristics cannot be verified, then the component must be designed and manufactured under a 10 CFR Part 50, Appendix B-compliant QA program.

ISG-06 states that a high-quality software development process that meets the 10 CFR Part 50, Appendix B, QA requirements is a required critical characteristic of all safety-related software. Verifying this critical characteristic as part of the commercial-grade dedication process for commercially developed software allows the software to be used in safety-related systems. ISG-06 references EPRI TR-106439 for guidance on evaluating and accepting commercial grade equipment. ISG-06 references the NRC safety evaluation of EPRI TR-106439, which states that V&V activities typical to a high-quality software development process is a critical characteristic that can be verified as part of the commercial-grade dedication process through the use of audits, examinations, and tests. An ISL observation, based on our analysis of this ISG-06 position for applicability to software tools, is that ISG-06 could be interpreted to mean that commercially developed software tools can be dedicated by verifying that V&V activities have been performed correctly by conducting audits, examinations, and tests as part of the commercial-grade dedication process.

The acceptance of software for safety system functions is based upon conclusions derived from: (1) evidence that acceptable software development plans were prepared and followed and (2) evidence that the software development process produced acceptable design outputs. The technical evaluation performed as part of the review and approval process verifies compliance with the requirements of IEEE 603-1991 (invoked by 10 CFR 50.55a(h)) and IEEE 7-4.3.2-2003 (endorsed by RG 1.152).

The bulk of the discussion in ISG-06 describes twelve review areas addressing different aspects of the licensing review. Some areas focus on the integrated system while other areas focus on single aspects of the digital safety system review. The review areas relevant to software tools involve the software architecture, software development process, and conformance with IEEE 7-4.3.2-2003 requirements.

ISG-06 states that the NRC review of the software architecture provides a high-level understanding of how the high-level coded functions interact to accomplish the safety functions. This high-level review facilitates subsequent detailed reviews and evaluations. ISG-06 states that when software is used to generate a hardware layout to be implemented in a FPGA, the review of the software tools used to generate, implement, and maintain the FPGA should be

guided by IEEE 7-4.3.2-2003, Clause 5.3.2. ISG-06 also reiterates the EPRI TR-106439 guidance that operating history alone is not sufficient for commercial dedication of equipment and that operating history (Acceptance Method 4 of EPRI NP-5652 and TR-102260) should be used in combination with one or more of the other three commercial dedication approaches (i.e., special tests and inspections (EPRI Method 1), source evaluations (EPRI Method 3), and supplier commercial-grade surveys (EPRI Method 2)). The acceptance criteria for the software architecture description are contained in branch technical position (BTP) 7-14, Section B.3.3.2. Note that NRC Generic Letter (GL) 89-02 places restrictions on the use of EPRI Method 4 alone, and the NRC has historically favored using the most appropriate and effective acceptance method(s) for each critical characteristic in a dedication process as reflected in NRC vendor inspection reports (See NUREG-0040) and procedures.

ISG-06 states that the NRC review of the software development process should focus on whether the software development activities and plans are sufficiently detailed to comply with applicable standards and acceptance criteria. ISG-06 provides important insight and guidance for the review process by referencing and discussing the acceptance criteria of BTP 7-14. The discussion and review guidance is organized consistent with the content of BTP 7-14 including major categories of software planning documentation, software plan implementation, and design outputs. The ISG-06 guidance for the software planning documentation covers each of the twelve software development plans covered by BTP 7-14. Guidance for the review of and acceptance criteria for software tools is covered in four software planning documents: software development plan, software integration plan, software V&V plan, and the SCM plan.

ISG-06 states that the software development plan should discuss the evaluation of methods and tools used. Of particular interest is the method by which the output of software tools is verified to be correct. ISG-06 reiterates the IEEE 7-4.3.2-2003 requirement that software tools should be designed as safety-related unless the tools are used in a manner such that defects not detected by the tool are detected by V&V activities. An ISL observation, based on our analysis of the ISG-06 review guidance, is that although ISG-06 provides some insight into the content and acceptance criteria of the software development plan, ISG-06 does not provide any additional software tool guidance beyond what is identified in BTP 7-14. ISL further observes that the overall treatment of software tools is based on endorsement of old standards. Specifically, ISG-06 and BTP 7-14 do not acknowledge the changes in software tool acceptance criteria in IEEE 7-4.3.2-2010 and the latest regulatory guides do not endorse IEEE 7-4.3.2-2010. As the IEEE is currently developing an update to IEEE 7-4.3.2-2010, and the NRC is represented on the cognizant IEEE subcommittee, there is an opportunity to influence the revision to better address software tools, among other issues important to the NRC, and to enhance the standard's ability to be endorsed by the NRC.

ISG-06 states that the software integration plan should discuss the tools, techniques, and methodologies to perform the software integration. ISG-06 reiterates the BTP 7-14 position that the software integration plan should require integration tools to be qualified with a degree of rigor and a level of detail appropriate to the safety significance of the software being created. An ISL observation, based on our analysis of ISG-06 review guidance, is that although ISG-06

provides some insight into the content and acceptance criteria of the software integration plan, ISG-06 does not provide any software tool guidance beyond what is identified in BTP 7-14.

ISG-06 states that the V&V reports should summarize the actions performed and the methods and tools used. An ISL observation, based on our analysis of ISG-06 review guidance, is that although ISG-06 provides some insight into the content and acceptance criteria of the V&V plan and reports, ISG-06 does not provide any software tool guidance beyond what is identified in BTP 7-14. ISL also observes that neither ISG-06 nor BTP 7-14 require V&V tools to be qualified even though the tools are used to develop safety-related software.

ISG-06 states that a critical item that should be discussed in the SCM plan is what items should be under configuration control. ISG-06 states that all design inputs and products, including software, should be controlled. Of particular importance is any software or software information that affects the safety software such as support software used in development, testing software, and tools used for management, development, or assurance tasks because each of these can affect the safety-related software if a wrong version is used during the development process. ISG-06 also acknowledges that many CM tools are available and that any tool used for this purpose should be selected, evaluated, and used properly. An ISL observation, based on our analysis of this ISG-06 discussion, is that although ISG-06 is consistent with BTP 7-14, the discussion adds no additional information.

The remaining discussion of software tools in ISG-06 is related to reviews evaluating conformance to the requirements of IEEE 7-4.3.2-2003. ISG-06 states that the licensee must provide a software QA plan to meet the requirements of BTP 7-14. ISG-06 states that software tools used to program the system, maintain configuration control, or track criteria for the requirement traceability matrix should be discussed in the QA plan even though these tools are not resident at run time and are generally not safety-related. The QA plan should also describe the method used to determine that these software tools were suitable for use. An ISL observation, based on our analysis of this review guidance, is that although the guidance recognizes that some software tools do not require the same degree of QA as safety-related software, no specific guidance is provided that specifies the appropriate degree of QA needed for these types of tools.

ISG-06 reiterates the requirements of IEEE 7-4.3.2-2003, Clause 5.3.2, that specify that tools must be developed to a similar standard as the safety-related software or used in a manner such that defects not detected by the tools are detected by V&V activities. ISG-06 also reiterates the requirements of BTP 7-14 which require the software development plan to identify tools and specify that tools should be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software developed using the tools. ISG-06 requires reviewers to evaluate tool usage in the overall context of the quality control and V&V processes to conclude that the tools function as described. An ISL observation, based on our analysis of this ISG-06 guidance, is that although the guidance is consistent with BTP 7-14 and accurately states the IEEE 7-4.3.2-2003 requirements, ISG-06 provides no additional guidance for the review of software tools other than the use of the tools may be audited to assess the quality of the tools.

2.3.1.9 NUREG-0800 Branch Technical Position 7-14

BTP 7-14 provides the acceptance criteria for review of the structure and content of documentation associated with the software development processes of safety-related software to ensure that the DI&C safety systems are designed, fabricated, installed, and tested to quality standards commensurate with the importance of the safety functions to be performed. Implementation of an acceptable software life cycle model, with appropriate processes and activities, provides the necessary software quality. The review of DI&C safety systems emphasizes quality, diversity, and defense-in-depth and is subdivided into three topics: planning, implementation, and design outputs. The review of each of these three topics is intended to verify that safety is maintained throughout the development of high-quality software. Software tools are only discussed in the context of planning documents; therefore, the following discussion and analysis focuses on the review guidance and acceptance criteria of planning documents.

The BTP 7-14 discussion of planning contains acceptance criteria of required content and characteristics of various plans that are part of the software development life cycle. The acceptance criteria are divided into three sets: management characteristics, implementation characteristics, and resource characteristics. Management characteristics involve aspects of planning that are significant to the management of project activities described in the planning documents. Implementation characteristics involve aspects of planning that describe the work necessary to achieve the purpose of the planning documents. Resource characteristics involve aspects of planning that describe the material resources necessary to perform the work defined in the planning documents. BTP 7-14 characterizes methods and tools used to develop safety system software as resource characteristics. A general acceptance criterion for methods and tools is that tools should be developed or dedicated as safety-related if the output of the tool cannot be proven to be correct, such as may occur if the tool produces machine language software code.

BTP 7-14 states that the software management plan should identify the tools required to perform project management including office equipment, computer hardware, and computer software. BTP 7-14 does not discuss project management tool qualification or provide any review guidance for project management tools. An ISL observation, based on our analysis, is that the BTP 7-14 position of not addressing project management tool qualification implies that project management tools are not considered to be safety-related. This observation is supported by the ISG-06 statement that software tools used to program the system, maintain configuration control, or track criteria for the requirement traceability matrix are not resident at run time and are generally not safety-related.

BTP 7-14 states that the software development plan should describe tools to be used during software development and should identify suitable tools to facilitate the production, management, and publication of documentation supporting the development of the software. The software development plan should further describe the software development environment, including software design aids, compilers, loaders, and subroutine libraries. The software development plan should also require that tools be qualified with a degree of rigor and level of

detail appropriate to the safety significance of the software that is to be developed using the tools. Methods, techniques, and tools that produce results that cannot be verified to an acceptable degree or that are not compatible with safety requirements should be prohibited, unless analysis shows that the alternative would be less safe. BTP 7-14 provides review guidance for the software development plan. The review guidance states that tools should be evaluated, especially the method by which the output of the tool will be verified to be correct. BTP 7-14 summarizes the acceptance criteria from IEEE 7-4.3.2-2003 by stating that software tools should be used in a manner such that defects not detected by the software tool will be detected by V&V activities. If defects cannot be detected by V&V activities, then BTP 7-14 states that the tool itself should be safety-related.

An ISL observation, based on our analysis of the software development plan acceptance criteria and review guidance, is that BTP 7-14 requires software development tools to be qualified or dedicated as safety-related, but provides no specific details on the level of rigor necessary to perform the qualification or dedication. Also, BTP 7-14 incompletely summarizes the requirements of IEEE 7-4.3.2-2003 in that BTP 7-14 does not recognize that tool operating experience can be used to provide additional confidence in the tool or that a test tool validation program can be used to provide tool confidence. The analysis also notes that BTP 7-14 does not acknowledge that the software tool requirements of IEEE 7-4.3.2-2003 have been updated in IEEE 7-4.3.2-2010. The software tool requirements in IEEE 7-4.3.2-2010 require that the software tool output be verified and validated to the same level of rigor as the safety-related software or that the tool be developed to the same level of quality as the safety-related software or be commercially dedicated.

BTP 7-14 states that the software QA plan should specify QA personnel participation in the assessment and review of project-specific tools and should identify any tools that will be used to facilitate the QA work including computer equipment and software. BTP 7-14 does not discuss QA tool qualification or provide any review guidance for QA tools. An ISL observation, based on our analysis, is that QA and project management tools only need to be identified in plans while the software development plan should require software development tools to be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software that is to be developed using the tools. This implies a different safety significance of QA and project management tools relative to software development tools.

BTP 7-14 states that the software integration plan and software installation plan should require that integration and installation tools be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software that is created or installed by the tools. An ISL observation, based on our analysis, is that integration and installation tools are treated similarly to software development tools in that the associated plans should require the tools to be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software. This is different from QA and project management tools that only need to be identified in plans. Although BTP 7-14 states that the integration and installation plan should require that integration and installation tools be qualified, BTP 7-14 does not identify any document that can be used to guide and tailor the qualification activities to the safety significance of the software.

BTP 7-14 states that the software maintenance plan should describe the tools necessary to accomplish the maintenance function. BTP 7-14 also states that the software maintenance plan should require that maintenance tools be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software that is created by the tools. BTP 7-14 also states that the software maintenance plan should require maintenance tools to be identical to tools used during the original design and should provide for the possible need to qualify a new version of the tool if the original version of the tool is no longer available. An ISL observation, based on our analysis, is that tools necessary to accomplish the maintenance function are treated in the same manner as software development tools and software integration and installation tools in that the associated plans should require those tools to be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software. This is different from QA and project management tools that only need to be identified in plans. Although BTP 7-14 states that the maintenance plan should require maintenance tools to be qualified and requires that the level of rigor necessary to perform the qualification be consistent with the safety significance of the software, BTP 7-14 does not identify any document that can be used to guide and tailor the qualification activities to the safety significance of the software.

BTP 7-14 states that the software training plan should describe any tools that will be used to facilitate training. BTP 7-14 states that training should be conducted on a training system equivalent to the actual hardware and software system. BTP 7-14 states that reviewers should take the same training as developers so that the reviewer can assess the adequacy of the training as well as become familiar with the tools used by the developers. An ISL observation, based on our analysis of the software training plan acceptance criteria, is that BTP 7-14 does not discuss that the software training plan should require training tool qualification nor does it discuss any review guidance for training tools similar to project management tools and QA tools.

BTP 7-14 states that the software operating plan should describe any tools that will be used to operate the software. BTP 7-14 does not discuss operating tool qualification or provide any review guidance for tools used to operate the software similar to project management tools, QA tools, and software training tools. An ISL observation, based on our analysis of the software operating plan is that the treatment of operating tools is different from software development tools, integration and installation tools, and maintenance tools in that operating tools are not required to be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software. Because tools that operate safety-related software could impact that software, ISL recommends that the NRC consider modifying BTP 7-14 to state that the software operating plan should require that tools used to operate the software should be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software that is operated by the tools. Note that tools used to operate or control safety-related software, to the extent that they could adversely affect the software, are an example of 10 CFR 50.49(b)(2) items. Although not electrical equipment per se, they could be viewed as associated nonsafety-related items, in effect a system or equipment component, whose failure or malfunction could impact one or more safety functions, in this case, by adversely affecting the functionality of the safety-related software. Therefore under 10 CFR 50.49(b), they would be deemed important to safety.

BTP 7-14 states that the software safety plan should specify the person or group responsible for each software safety task and that a designated safety officer should have clear authority to reject the use of a tool if it cannot be shown that the tool will not impact the safety of the final software system. BTP 7-14 also states that the software safety plan should describe the tools used to perform safety activities and should specify a process for selecting tools. The software safety plan should also describe methods for preventing inadvertent introduction of hazards by the use of project tools. BTP 7-14 does not discuss the qualification or provide any review guidance for tools used to perform safety activities. An ISL observation, based on our analysis of the software safety plan, is that BTP 7-14 considers these types of tools to be similar to project management tools, QA tools, training tools, and tools that operate on software in that they are not required to be qualified.

BTP 7-14 states that the software V&V plan should describe V&V reporting requirements and that the content of the reports should include a summary of the tools used. BTP 7-14 also states that the software V&V plan should specify a process for selecting V&V tools and should describe the hardware and software environment and necessary controls applicable to the V&V tools. BTP 7-14 does not discuss the qualification requirements or provide any review guidance for V&V tools. An ISL observation, based on our analysis of the software V&V plan is that the treatment of V&V tools is consistent with the treatment of project management tools, QA tools, training tools, tools that operate on software, and tools used to perform safety activities. Based on our analysis, ISL recommends that the NRC consider modifying BTP 7-14 to state that the V&V plan should require V&V tools to be qualified or commercially dedicated since the tools may fail to detect defects in the safety-related software. However, the level of rigor associated with V&V tool qualification should be less than the rigor associated with qualifying or dedicating tools used in the design and implementation phases of safety-related software development.

BTP 7-14 states that the SCM plan should require configuration control of support software, including tools, used in the software project for management, development, and assurance tasks. The SCM plan should also describe the process used to maintain and track software tools purchased and used during software development. The SCM plan should provide a method of tracking tool history, defects, and errata sheets to determine which outputs are affected by which tool defect. BTP 7-14 also states that the SCM plan should identify suitable tools used in the performance of CM activities, specify a process for selecting the suitable tools, and describing the hardware and software environment and necessary controls for executing the tools. BTP 7-14 does not discuss the qualification or verification requirements or provide any review guidance for CM tools. Based on our analysis, ISL recommends that the NRC consider modifying BTP 7-14 to state that the SCM plan should require some form of qualification or verification of CM tools since a configuration error could result in the wrong version of the code or library to be compiled into the final executable even though V&V testing of the final software should detect that a CM error introduced a defect into the final software.

BTP 7-14 states that the software test plan should identify tools used for testing, specify a process for selecting testing tools, and describe the hardware and software environment and necessary controls for executing the testing tools. BTP 7-14 does not discuss the qualification or verification requirements or provide any review guidance for testing tools. Based on our

analysis, ISL recommends that the NRC consider modifying BTP 7-14 to state that the software test plan should require some form of qualification or verification of testing tools since an error in the tool could result in the failure to detect a real defect or false detection of a non-existing defect in the safety-related software.

An ISL observation, based on our analysis of BTP 7-14, is that software tools are defined as resource characteristics of planning documents and the review guidance and acceptance criteria for planning documents is discussed in terms of planning documents produced as part of a well-organized software development life cycle process. Planning documents for most phases of the software development life cycle are required to describe tools used in that life cycle phase and to define the process by which tools are selected. Planning documents are also required to define the tool operational environment and any special controls necessary to execute the tools. In some cases, the planning document requires a tool qualification plan for tools used in a phase. BTP 7-14 does not state that all plans should require tool qualification. Tools that are required to be qualified are done so with a degree of rigor and level of detail appropriate to the safety significance of the software developed with the tools; however, BTP 7-14 does not provide any specific guidance for qualifying the tool. BTP 7-14 also references software tool requirements from IEEE 7-4.3.2-2003 which have been superseded by IEEE 7-4.3.2-2010. Based on our analysis, ISL recommends that the NRC consider modifying BTP 7-14 for consistency with software tool guidance in IEEE 7-4.3.2-2010 and BTP 7-14 should state that plans should require software tool qualification in more life cycle phases.

Another ISL observation, based on our analysis of BTP 7-14, is that the review guidance and acceptance criteria of planning documents for developing safety-related software should be extended to the planning, implementation, and design outputs of software tools when the tools could impact the final safety-related software product. Tool documentation, implementation, and design outputs should be reviewed with the same rigor as the software they are designed to produce. For COTS tools, the planning activities and products may not be available for review; therefore, the COTS tool review and acceptance criteria should focus on the implementation and design outputs of the COTS tool.

2.3.1.10 *NUREG-0800 Appendix 7.1-D*

NUREG-0800, Appendix 7.1-D, provides guidance for evaluating the application of the safety system criteria in IEEE 7-4.3.2-2003 to computer-based safety systems and provides guidance for evaluating the application of cyber security criteria in RG 1.152 to develop and maintain a secure development and operational environment for using digital computers in the safety systems of nuclear power plants. IEEE 7-4.3.2-2003 provides guidance on applying the safety system criteria to computer-based safety systems and is endorsed by RG 1.152, Revision 3. IEEE 7-4.3.2-2003 specifies computer-specific criteria (including software) to supplement the criteria in IEEE 603-1998. Although IEEE 7-4.3.2-2003 references IEEE 603-1998 and IEEE 7-4.3.2-2010 references IEEE 603-2009, IEEE 603-1991 and the correction sheet dated January 30, 1995 remain the requirement for safety systems in accordance with 10 CFR 50.55a(h). The criteria contained in IEEE 7-4.3.2-2003, in conjunction with requirements in IEEE 603-1991, establish minimum functional and design criteria for computers used as components of a safety

system. Although intended for digital safety systems, the criteria of IEEE 7-4.3.2-2003 can be applied to any DI&C system. For nonsafety DI&C systems that have a high degree of importance-to-safety based on risk, Appendix 7.1-D recommends that a graded application of the criteria of IEEE 7-4.3.2-2003 should be considered. This graded approach is not discussed in IEEE 7-4.3.2-2003; however, guidance on use of a graded approach can be found in EPRI TR-102260, that may be applied throughout the CGI dedication process that meets the intent of 10 CFR Part 50, Appendix B, in which Criterion II states that the QA program shall control activities affecting the quality of structures, systems, and components (SSCs) consistent with their importance to safety.

Appendix 7.1-D discusses concerns of digital computer-based systems related to common-cause failures. Common-cause failure is a concern when a system design uses shared databases or shared process equipment because a software programming error can defeat the redundancy achieved by the hardware architectural structure. An ISL observation, based on our analysis of this concern over common-cause failures, is that even though software tools are not discussed in relation to common-cause failures, the NRC should consider regulatory guidance recommending that special care be taken when reviewing software tool usage, especially compilers and linkers, to ensure that the single-failure criterion is not violated. For example, diverse systems that have been compiled with a single compiler may subject the code output to a common-mode failure mechanism.

Appendix 7.1-D discusses software tools in the context of QA. Appendix 7.1-D states that software quality is addressed in IEEE 12207.0-1996 which is a predecessor to the current version, IEEE 12207-2008, and software tools are addressed in IEC 60880-2 Ed 1.0 which is a predecessor to the current version, IEC 60880 Edition 2.0. Appendix 7.1-D reiterates the position of these standards that if defects not detected by software tools or introduced by software tools will not be detected by V&V activities, then the tool itself should be designed as safety-related software in accordance with all the requirements of 10 CFR Part 50, Appendix B. Appendix 7.1-D also reiterates the IEEE 7-4.3.2-2003 requirements that software tools used to support software development processes and V&V processes should be controlled under CM and that either a test tool validation program should be developed or tools should be used in a manner such that defects not detected by the tool will be detected by V&V activities to confirm that software tools are suitable for use in the development of safety-related software. Appendix 7.1-D also reiterates that tool-operating history may be used to provide additional confidence in the suitability of tool. The Appendix 7.1-D discussion of software tools continues by reiterating the BTP 7-14 requirement for a thorough discussion and identification of software tools in all software development life cycle planning documents and that most tools should be qualified with a degree of rigor and level of detail commensurate with the safety significance of the software developed using the tool. The Appendix 7.1-D discussion of software tools concludes by reiterating that reviewers should thoroughly evaluate tool usage as part of the overall quality control process and that testing alone can only show that the tool operates as intended and cannot be relied upon to show that no unintended functions exist.

An ISL observation, based on our analysis of Appendix 7.1-D, is that no new guidance is provided that is not already covered by the referenced documents and that Appendix 7.1-D

simply stresses the importance of software quality and software tool evaluations. Since all of the documents referenced by Appendix 7.1-D have been superseded, ISL recommends, based on our analysis, that the NRC consider modifying Appendix 7.1-D and 10 CFR 50.55a(h) for consistency with the latest documents that address the significant evolution of safety-related DI&C software development and software tools used in the development processes.

2.3.1.11 NUREG/CR-6263

NUREG/CR-6263 examines the technical basis of guidelines that could be considered for reviewing and evaluating high-integrity computer software for use in safety systems of nuclear power plants. NUREG/CR-6263 discusses COTS and pre-developed software, including compilers, only within the context of the development of new, safety-related software.

NUREG/CR-6263, Volume 1, Executive Summary, Table ES-1, Section 5, contains a discussion of the development environment, including software tools, as part of the software coding process. Guidelines for the development environment are grouped into seven areas. Four areas are associated with a research need to provide acceptance criteria for evaluating the adequacy of the software development environment. The remaining guidelines specify that:

- (1) tools should have proven vendor support,
- (2) tools should be kept under CM,
- (3) high-level languages should be used, and
- (4) detailed programming standards should exist for each language used.

These guidelines would be beneficial if incorporated into any new software tool regulatory guidance; however, the requirement that tools have proven vendor support is difficult to quantify. An ISL observation, based on our analysis of this technical requirement, is that vendor support should be proven through formal, written support agreements or historical records of prior vendor support. Vendor support should also be augmented with comprehensive tool documentation and accessibility to tool bug reports and fixes.

NUREG/CR-6263, Volume 2, Sections 5.1 and 5.2, amplify the discussion in Volume 1, Table ES-1, with additional guidelines for the development environment. The additional guidelines recommend:

- (1) using the safety analysis of the system being developed to determine the extent of the tool evaluation,
- (2) analyzing the types of errors that might be introduced through the use of a tool or set of tools, and
- (3) using an external certification process, if one exists.

ISL's analysis of the portion of NUREG/CR 6263 relating to software tools suggests that the guidelines are reasonable but implementation details for most areas were left for future research.

2.3.1.12 NUREG/CR-6421

NUREG/CR-6421 is a report on a proposed acceptance process for COTS software in reactor operations. This report proposes determining the acceptability of COTS software, including software tools, using a classification scheme based on the importance to safety of the system in which the product will be used. It recommends that software tools be rated for safety impact, initially evaluated using assurance methods for similar tools, and continuously evaluated during tool use.

A goal of this report is to set the recommended number of safety categories for COTS products large enough to enable them to be applied efficiently and small enough to prevent fragmentation. The suggested number of categories is larger than two (safety and nonsafety) and less than five. The suggested safety categories of A, B, C, and unclassified are derived from a predecessor to IEC 61226 and RG 1.97. Software tools could be placed in any one of the four categories depending on the tool function and the associated software classification. If a software tool can embed an error in other software important to safety, then the software tool is assigned the same category as the software in which the error can occur. If the software tool can only challenge software important to safety (e.g., exposing existing errors), then the software tool is assigned the next lower safety category.

Table 3 of NUREG/CR-6421 defines four COTS safety category criteria depending on how the COTS product is used. These criteria, included below, are used to determine the safety category of the product being evaluated.

- Criterion 1 If the COTS product is used directly in a system important to safety, the COTS safety category is determined by the criteria of IEC 61226.
- Criterion 2 If the COTS product directly produces or controls the configuration of an executable software product that is used in a system important to safety, and no method exists to validate the output of the COTS product, the COTS safety category is the same as that of its output, except that category C software may be produced by COTS products of the unclassified category. COTS software that directly produces category A or B software that is validated by other means is category B or C, respectively.
- Criterion 3 If the COTS product supports production of category A, B, or C software, but does not directly produce or control the configuration of such software modules, it falls under the safety category "unclassified".
- Criterion 4 If the COTS product has no impact on category A, B, or C software or systems, it falls under the safety category "unclassified".

Software tools fall into criteria descriptions 2, 3 or 4 depending on how the tool is used and the safety category of the system developed or tested using the tool.

The proposed acceptance process starts with a risk and hazards analysis for using the tool for a specific purpose in a defined environment. All tools are initially evaluated. Only tools that either directly produce or control the configuration of an executable product that is used in a system important to safety are recommended for further evaluation. For those products, the safety category is then assessed and this determines the rigor of the acceptance criteria.

NUREG/CR-6421 provides useful guidance for determining if a software tool requires certification, meaning NRC approval in this context. According to this report, many software tools can be generically approved. For those that are determined to be safety significant, Tables 5 through 7 of NUREG/CR-6421 contain detailed acceptance criteria.

The usefulness of this document is limited by its age and therefore the reliance on standards that have since been amended or superseded. However, it describes a path that can be followed to assess and accept a software tool.

2.3.2 Summary

NRC documents provide high-level software tool guidance primarily through endorsement or adoption of IEEE standards. Regulatory guidance in the areas of software tool commercial-grade dedication, qualification, development, review, and approval could be improved.

RGs 1.152 and 1.168 through 1.173 provide some software tool guidance, primarily through endorsements of various IEEE standards. RG 1.152 focuses on establishing and maintaining a secure operational environment and a secure development environment for system software. An ISL observation, based on our analysis, is that this guidance does not apply to software tools since the operational environment of the software is not relevant to the software tools used to develop the software and the development environment of the software tool does not need to be secure to develop safety-related software for the reasons stated above. RG 1.152 also requires that nonsafety-related software executed on the same computer as safety-related software be treated as part of the safety system.

RG 1.168 endorses IEEE 1012-2004 which uses a four-level safety integrity scheme (SIL 1 through SIL 4) for quantifying software criticality; however, RG 1.168 states that all software developed for safety systems should be classified as SIL 4. Because RG 1.152 requires that nonsafety software be classified as part of the safety system and RG 1.168 assigns an SIL 4 to safety systems, then even nonsafety system software is assigned an SIL 4 subject to the most rigorous V&V activities. An ISL observation, based on our analysis of these RGs, is that this situation is inconsistent with 10 CFR Part 50, Appendix B, Criterion II, which states that the QA program shall provide control over activities affecting the quality of the identified SSCs, to an extent consistent with their importance to safety. Thus, the QA program controlling the software development activities should be applied consistent with the software's importance to safety. RG 1.168 also states that tools used in the development of safety system software should meet the V&V requirements of IEEE 7-4.3.2-2003 when the V&V activities cannot detect flaws in the software produced by the tools.

RG 1.169 endorses IEEE 828-2005 for planning and executing a SCM program. RG 1.169 states that software support tools used in the development and testing of safety system software must be handled in accordance with the requirements of IEEE 7-4.3.2-2003 and identified and controlled as configuration items or discussed in controlled documents. RG 1.169 also states that items that may not change but are necessary to ensure correct software production, such as compilers, should also be configuration items.

RG 1.170 endorses IEEE 829-2008 for requirements associated with the form and content of documentation of safety system software testing. IEEE 829-2008 does not require repetition of test information in test documentation if the test information is managed by an automated tool. RG 1.170 takes exception to this requirement because the repetition of test information is beneficial to make test information easily accessible to improve the timeliness of reviews.

RG 1.171 endorses IEEE 1008-1987 for software unit testing requirements. IEEE 1008-1987 requires the use of the test design specifications and the test summary report documents defined in IEEE 829-2008 with a requirement for additional information to cover unit testing documentation. RG 1.171 states that the documentation used to support software unit testing must include, as a minimum, environmental conditions and special controls, equipment, tools, and instrumentation needed for the accomplishment of testing.

RG 1.172 endorses IEEE 830-1998 for requirements related to the software requirements specification. IEEE 830-1998 discusses the use of specification or representation tools to generate requirements including a requirement to retain the natural language descriptions for better understanding of the software requirements specification. RG 1.172 reinforces the IEEE 830-1998 requirements by stating that traceability should be maintained between representations generated by tools and the natural language descriptions of the software requirements that are derived from systems requirements and system safety analyses. An ISL observation, based on our analysis of RG 1.172, is that the NRC should consider modifying RG 1.172 or influencing the IEEE to modify IEEE 830-1998 to require the documentation of detailed information on any tool used to develop a software requirements specification (e.g., acquisition, training, support, and qualification) because tool information is not required to be documented by IEEE 830-1998. RG 1.172 should also be updated to endorse IEEE 29148-2011 which supersedes IEEE 830-1998.

RG 1.173 does not discuss software tools and simply endorses IEEE 1074-2006 for defining a set of software life cycle processes and activities appropriate for the development of safety system software. IEEE 1074-2006 is a standard for developing software life cycle processes but does not address system-level issues and the overall system-level development processes must be developed in accordance with other regulations or industry standards. An ISL observation, based on our analysis, is that the NRC should consider revising RG 1.173 to endorse IEEE 12207-2008 and IEEE 15288-2008 which contain software and system development processes, respectively, that have IEEE, ISO, and IEC endorsement.

ISG-06 provides guidance for the NRC review of license amendments supporting installation of DI&C equipment in operating nuclear power plants originally licensed under 10 CFR Part 50. ISG-06 also identifies the documentation that should be reviewed and when the documentation

should be provided. Consistent with 10 CFR Part 21, ISG-06 recognizes two methods of approving a component for use in safety-related applications (i.e., a “basic component”). The first method is to dedicate a CGI and the second method is to design and manufacture the component under a 10 CFR Part 50, Appendix B-compliant QA program. Software tools are not mentioned in ISG-06 with respect to the approval processes; however, the two approval methods could be applied to software tools. Specifically, ISG-06 could be interpreted to mean that commercially developed software tools can be dedicated by verifying that V&V activities have been performed correctly by conducting audits, examinations, and tests as part of the commercial-grade dedication process. ISG-06 also requires that a license amendment request address all the criteria associated with the software development process and conformance with IEEE 7-4.3.2-2003 software tool requirements. The remaining discussions in ISG-06 provide important insight and guidance for the review process by referencing and discussing the acceptance criteria of BTP 7-14.

BTP 7-14 provides the acceptance criteria for review of the structure and content of documentation associated with the software development process of safety-related software. Software tools are only discussed in the context of planning documents. BTP 7-14 characterizes methods and tools used to develop safety system software as resource characteristics. A general acceptance criterion for methods and tools is that tools should be developed or dedicated as safety-related if the output of the tool cannot be proven to be correct. Planning documents for most phases of the software development life cycle are required to describe tools used in that life cycle phase and to define the process by which tools are selected. Planning documents are also required to define the tool operational environment and any special controls necessary to execute the tools. In some cases, the planning document requires a tool qualification plan for tools used in a phase. BTP 7-14 does not require all plans to require software tool qualification. Tools that are required to be qualified are done so with a degree of rigor and level of detail appropriate to the safety significance of the software that is developed with the tools; however, BTP 7-14 does not provide any specific guidance for qualifying the tool. BTP 7-14 also references software tool requirements from IEEE 7-4.3.2-2003 which has been superseded by IEEE 7-4.3.2-2010. Based on our analysis, ISL recommends that the NRC consider updating BTP 7-14 for consistency with software tool guidance in IEEE 7-4.3.2-2010 and should require software tool qualification in more life cycle phase planning documents. Again, BTP 7-14 uses the term “tool qualification” in the same sense as RTCA DO-330. Tool qualification activities would form part of the review for suitability of application that is required by Criterion III of Appendix B to 10 CFR Part 50 and the acceptance requirements of Criterion VII of Appendix B to Part 50, or it would be an integral part of the commercial-grade dedication process for tools not developed under an Appendix B QA program, which, when conducted as prescribed in 10 CFR Part 21, is deemed to be an alternate and equivalent means of meeting the requirements of Appendix B to 10 CFR Part 50, in particular, Criteria III and VII requirements.

NUREG-0800, Appendix 7.1-D, provides guidance for evaluating the application of the safety system criteria in IEEE 7-4.3.2-2003 to computer-based safety systems. The criteria contained in IEEE 7-4.3.2-2003, in conjunction with requirements in IEEE 603-1991, establish minimum functional and design criteria for computers used as components of a safety system. Appendix

7.1-D discusses software tools in the context of QA and reiterates the requirement that if defects not detected by software tools or introduced by software tools will not be detected by V&V activities, then the tool itself should be designed as safety-related software. An ISL observation, based on our analysis of Appendix 7.1-D, is that no new guidance is provided that is not already covered by the referenced documents and that Appendix 7.1-D simply stresses the importance of software quality and software tool evaluations. Since all of the documents referenced by Appendix 7.1-D have been superseded, ISL recommends that the NRC consider updating Appendix 7.1-D and 10 CFR 50.55a(h) for consistency with the latest documents that address the significant evolution of safety-related DI&C software development and software tools used in the development processes.

NUREG/CR-6263 discusses guidelines for evaluating the integrity of computer software for use in safety systems of nuclear power plants. Software tools are discussed in the context of the overall software development environment and the software coding process. NUREG/CR-6263 guidelines for software tools covers vendor support of the tool, CM of the tool, and the use of high-level languages with detailed programming standards. NUREG/CR-6263 also recommends that the system safety analysis should determine the extent of tool evaluations, an error analysis should determine the types of errors that can be introduced by the tool(s), and an external certification process should be used to certify the tool(s). The usefulness of this document is limited by its age and the lack of implementation details.

NUREG/CR-6421 contains an acceptance process for COTS software in reactor operations that determines the acceptability of COTS software, including software tools, using a classification scheme based on the importance to safety of the system in which the product will be used. NUREG/CR-6421 defines four safety category criteria depending on how the COTS product is used which is inconsistent with the categories of systems important to safety in 10 CFR 50.49(b) (i.e., safety-related, nonsafety-related, and post-accident monitoring systems). The usefulness of this document is limited by its age and therefore the reliance on standards that have since been amended or superseded. However, it does describe a path that can be followed to assess and accept a software tool.

NRC RGs, SRP BTPs and appendices, ISGs, and contractor reports endorse IEEE standards for software tool guidance. Based on the analysis of NRC documents, the following essential elements of the documents relative to software tools have been determined.

- (1) RG 1.152 takes exception to the IEEE 7-4.3.2-2003 position on nonsafety software and clarifies that any software providing nonsafety functions that resides on a computer providing a safety function must be classified as a part of the safety system.
- (2) The main focus of the regulatory positions in RG 1.152 is to provide specific guidance for establishing and maintaining a secure development and operational environment for the protection of digital safety systems throughout the design and development life cycle phases of the safety system.

-
- (3) RG 1.168 takes exception to the IEEE 1012-2004 graduated SIL classifications and states that software used in nuclear power plant safety systems should be assigned a SIL 4 classification.
 - (4) RG 1.168 states that tools used in the development of safety system software should meet the requirements of IEEE 7-4.3.2-2003; however, the provisions of RG 1.168 only apply to software tools when V&V activities cannot detect flaws in the software produced by the tools. If V&V activities can detect flaws introduced into the software produced by the tools, then the V&V tasks of witnessing, reviewing, and testing of the software tools are not required.
 - (5) RG 1.169 states that for safety system software, support software (i.e., tools) used in the development of safety system software and testing tools used to test safety system software must be identified and controlled as configuration items or discussed in controlled documents.
 - (6) RG 1.169 states that items that may not change but are necessary to ensure correct software production, such as compilers, should also be configuration items, thereby ensuring that all factors contributing to the executable software are understood.
 - (7) RG 1.169 states that tools used in the development of safety system software should be handled according to IEEE 7-4.3.2-2003.
 - (8) RG 1.170 states that the risk assessment scheme in IEEE 829-2008 and the use of four integrity levels for safety system software is unacceptable because of the potential catastrophic consequences of safety system software failure.
 - (9) RG 1.170 takes exception to Clause 6.3 of IEEE 829-2008, which states that there is no need for repetition of test information in test documentation if the test information is completely managed by an automated tool, provided that the test documentation contains references for tracing the information.
 - (10) RG 1.171 states that the documentation to support software unit testing should include, as a minimum, special conditions and controls, equipment, tools, and instrumentation needed for the accomplishment of testing.
 - (11) RG 1.172 reinforces the IEEE 830-1998 requirements by stating that traceability should be maintained between representations generated by tools and the natural language descriptions of the software requirements that are derived from systems requirements and system safety analyses.
 - (12) ISG-06 recognizes two different methods of approving a component for use in safety-related applications: dedicate a CGI or design and manufacture the component under a 10 CFR Part 50, Appendix B-compliant QA program.
-

- (13) ISG-06 and BTP 7-14 do not acknowledge the changes in software tool acceptance criteria in IEEE 7-4.3.2-2010 or the proposed draft version of IEEE 7-4.3.2 and the latest regulatory guides do not endorse IEEE 7-4.3.2-2010.
- (14) BTP 7-14 defines software tools as resource characteristics of planning documents that are required to describe tools used, define the process by which tools are selected, and define the tool operational environment and any special controls necessary to execute the tools.
- (15) A general acceptance criterion for tools is that tools should be developed or dedicated as safety-related if the output of the tool cannot be proven to be correct, such as may occur if the tool produces machine language software code.
- (16) BTP 7-14 states that the software development plan, software integration plan, software installation plan, software maintenance plan, and should require tools in these life cycle phases to be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software developed using the tool. The software management plan, software QA plan, software training plan, software operating plan, software safety plan, software V&V plan, SCM plan, and software test plan are not required to discuss tool qualification.

Based on the analysis of NRC documents, ISL has made the following important observations and recommendations.

- (1) The barrier requirements of IEEE 7-4.3.2-2003 are not relevant to software tools and are only applicable to system hardware and system software.
- (2) Requiring equipment that performs both safety and nonsafety functions on the same computer to be classified as part of the safety system appears inconsistent with the classification and commercial-grade dedication of safety-related and nonsafety-related software based on their importance to safety as described in 10 CFR Part 50, Appendix B, and 10 CFR Part 21.
- (3) The NRC should consider updating RG 1.152 to address the IEEE 603-2009 changes to the classification and requirements of nonsafety-related systems.
- (4) The NRC should consider updating RG 1.152 to recognize the latest version of IEEE 7-4.3.2 that includes tool types when it is published.
- (5) Establishing and maintaining a secure operational environment and a secure development environment for system software is not relevant to software tools. Neither the operational environment nor development environment of the software tool needs to be secure even if the software tool is used to develop safety-related software because tools are not likely to be hacked while being developed and a would-be attacker of the application software would not know which tool was used or would be used in its development.

-
- (6) Classifying all safety-related software as SIL 4 does not allow V&V activities to be tailored based on the importance to safety of the various elements of the safety-related system as specified in 10 CFR Part 50, Appendix B, Criterion II.
 - (7) The NRC should consider expanding the guidance of RG 1.168 to cover all types of tools used in the software development process.
 - (8) The RISC-based categorization of SSCs in 10 CFR 50.69 is not recognized by RG 1.168 or RG 1.170.
 - (9) A recognized method of selectively applying software development requirements commensurate with the software's importance to safety is discussed in the EPRI TR-102260 commercial-grade dedication process.
 - (10) The NRC should consider modifying RG 1.172 or attempt to influence the IEEE to update IEEE 830-1998 to require the documentation of detailed information on any tool used to develop a software requirements specification (e.g., acquisition, training, support, and qualification) because tool information is not required to be documented by IEEE 830-1998.
 - (11) The NRC should consider updating the RG 1.172 endorsement of IEEE 830-1998 to IEEE 29148-2011 since IEEE 830-1998 has been superseded by IEEE 29148-2011 and IEEE 29148-2011 specifies the required processes that must be implemented for the engineering of requirements for systems and software products throughout the life cycle.
 - (12) The NRC should consider revising RG 1.173 to endorse IEEE 12207-2008 and IEEE 15288-2008, which contain software and system development processes, respectively, and have IEEE, ISO, and IEC endorsement.
 - (13) ISG-06 could be interpreted to mean that commercially developed software tools can be dedicated by verifying that V&V activities have been performed correctly by conducting audits, examinations, and tests as part of the commercial-grade dedication process.
 - (14) Because tools that operate safety-related software could impact that software, the NRC should consider modifying BTP 7-14 to state that the software operating plan should require that tools used to operate the software should be qualified.
 - (15) The NRC should consider modifying BTP 7-14 to state that the V&V plan should require V&V tools to be qualified or commercially dedicated since the tools may fail to detect defects in the safety-related software.
 - (16) The NRC should consider regulatory guidance recommending that special care be taken when reviewing software tool usage, especially compilers and linkers, to ensure that the single-failure criterion is not violated.
-

2.4 Institute of Electrical and Electronics Engineers Analysis

The Institute of Electrical and Electronics Engineers (IEEE) is the world's largest professional association dedicated to advancing technological innovation and excellence through its publications and technology standards. IEEE publishes nearly a third of the world's technical literature in electrical engineering, computer science, and electronics. With an active portfolio of nearly 1,300 standards, IEEE international standards are a central source for standardization in a broad range of emerging technologies including IT. NRC regulatory guides endorse several IEEE standards that contain requirements for promoting high functional reliability and establishing minimum functional requirements for computers used in nuclear power plants.

2.4.1 Document Analysis

The general philosophy of the IEEE speaks very little about software tools. When software tools are discussed, they are mentioned as tools that can be used as an alternative to manual means. IEEE documents cover few aspects on the use of software tools prior to 2003. Recent documents mention individual tools that can be used in preparing specific documentation and performing V&V activities. Other IEEE documents discuss concerns with common-cause failures. IEEE 7-4.3.2-2003, IEEE 7-4.3.2-2010, IEEE 1012-2004, and IEEE 1012-2012 contain software tool requirements and guidance to establish the suitability of software tools for developing safety-related systems. IEEE 14102-2010 provides guidance for the proper evaluation and selection of CASE tools. IEEE 603-2009 discusses design criteria of safety systems to deal with common-cause failures. The following paragraphs discuss the analysis of these standards with respect to the selection, use, verification, validation, qualification, review, and approval of software tools.

2.4.1.1 IEEE 7-4.3.2

The following discussion and analysis covers IEEE 7-4.3.2-2003, IEEE 7-4.3.2-2010, and a proposed draft version of IEEE 7-4.3.2. IEEE 7-4.3.2-2003 is the version currently endorsed by Regulatory Guide 1.152. Both IEEE 7-4.3.2-2003 and IEEE 7-4.3.2-2010 contain a few requirements for software tools but state that the standard does not address the justification for the selection of software tools and acceptance criteria for compilers, operating systems, and libraries. The proposed draft version of IEEE 7-4.3.2 adds requirements for the selection of tools and expands software tool requirements.

The proposed draft version of IEEE 7-4.3.2 requires that the benefits and risks associated with selecting software tools be taken into consideration and that the selected tools should limit the opportunity for making errors and introducing faults while maximizing the opportunity for detecting faults in the resulting safety-related programmable digital device (PDD). The proposed version also requires that the intended functionality and limitations of tools be identified and documented, tools shall not be used outside their documented functionality or limits without prior justification, software tools shall be assessed in a manner consistent with the category of tool and the potential of the tool to introduce faults into the safety-related PDD, and the assessment process for tools should take into account experience from prior use including

experience of developers as well as experience gained from processes in which the tools are used.

All three standards require that software tools be maintained under configuration control and state that validation of a tool is not necessary if the tool is 100% tested, meaning that testing shall include the application of every possible combination of inputs or input sets, if the function includes state-based logic, and the evaluation of all of the outputs that result from each combination of inputs. If 100% testing is not practical, then the testing that is excluded must be identified and the exclusion must be justified.

All three standards also require COTS equipment to be commercially dedicated to verify that the item dedicated is equivalent to basic components developed under an acceptable nuclear quality assurance program. The proposed draft of IEEE 7-4.3.2 adds a requirement that the dedication processes shall baseline all the critical functions of the COTS equipment such that the utility can perform accurate analyses on the consequences of unnecessary functions in the COTS system.

IEEE 7-4.3.2-2003, IEEE 7-4.3.2-2010, and the draft version of IEEE 7-4.3.2 require that software V&V satisfy the requirements for the highest safety integrity level (i.e., SIL 4) as defined in IEEE 1012-1998, IEEE 1012-2004, and IEEE 1012-2012, respectively. The three standards also specify slightly different methods to determine the suitability of software tools to be used in the development of safety-related systems.

IEEE 7-4.3.2-2003 requires that one or both of the following methods be used to confirm the suitability of software tools:

- a) A test tool validation program shall be developed to provide confidence that the tool functions as required.
- b) The software tool shall be used in a manner such that defects not detected by the tool will be detected by V&V activities.

IEEE 7-4.3.2-2010 requires that one or both of the following methods be used to confirm that outputs of software tools are suitable for use in safety-related systems:

- a) The output of the tool shall be subjected to the same level of V&V as the safety-related software, to determine that the output of that tool meets the requirements established during the previous life cycle phase.
- b) The tool shall be developed using the same or an equivalent high-quality life cycle process as required for the software upon which the tool is being used or commercially dedicated, to provide confidence that the necessary features of the software tool function as required.

The proposed draft version of IEEE 7-4.3.2 requires that one or both of the following methods be used to confirm outputs of that software tool are suitable for use in safety-related systems:

- a) The software tool shall be used in a manner such that defects not detected by the tool will be detected by V&V activities.
- b) The tool shall be developed using a quality life cycle process commensurate with the criticality of the safety functions performed by the end product.

An ISL observation, based on an analysis of the methods to determine the suitability of tools in the three versions of IEEE 7-4.3.2, is that any of the three sets of methods provides a reasonable path to confirm the suitability of tools. The IEEE 7-4.3.2-2003 method of developing a test tool validation program is similar to the IEEE 7-4.3.2-2010 method of subjecting the output of the tool to the same level of V&V as the safety-related software, to determine that the output of that tool meets the requirements established during the previous life cycle phase. Subjecting the output of the tool to the same level of V&V as the safety-related software is more rigorous. ISL considers both methods to be reasonable; however, the test tool validation program is a less rigorous and acceptable approach to determine the suitability of software tools. Neither of these methods are included in the draft version of IEEE 7-4.3.2.

IEEE 7-4.3.2-2003 and the proposed draft version of IEEE 7-4.3.2 both contain an identical option to determine the suitability of software tools. That method is that the tool shall be used in a manner that defects not detected by the tool will be detected by V&V activities is difficult to satisfy. ISL notes that this method does not exist in IEEE 7-4.3.2-2010 but was restored in the proposed draft version. ISL considers that an expectation that V&V activities will detect all defects not detected by the tool is not realistic and difficult to confirm.

IEEE 7-4.3.2-2010 and the proposed draft version of IEEE 7-4.3.2 contain similar options related to tool development using a high-quality life cycle process. IEEE 7-4.3.2-2010 requires that tools be developed using the same or an equivalent high-quality life cycle process as required for the software upon which the tool is being used or commercially dedicated. The proposed draft version of IEEE 7-4.3.2 does not mention dedication but requires that tools be developed using a quality life cycle process commensurate with the criticality of the safety functions performed by the end product. For COTS tools, which comprise the bulk of software development tools, the high-quality life cycle process development option is not usually viable since the COTS tools are typically developed for widespread use without regard for their application (i.e., nuclear safety-related software development). In this case, IEEE 7-4.3.2-2010 offers an option of commercially dedicating the COTS tool, whereas the proposed draft version of IEEE 7-4.3.2 does not offer this option. In accordance with the proposed draft version of IEEE 7-4.3.2, COTS tools would have to be used in a manner such that defects not detected by the tool will be detected by V&V activities. Based on an analysis, ISL considers that this option is difficult to confirm and is unrealistic.

The proposed draft version of IEEE 7-4.3.2 also adds five tool categories:

Type I – Software Development Tools – Type I tools are used by programmers for generating software/logic to be loaded into a computer system or PDD. Type I tools generate object code or logic synthesis that will be executed within the target system hardware.

Type II – Surveillance and Maintenance Tools – Type II tools are for surveillance testing, changing configuration data, and loading software or logic for safety systems. Type II tools may generate output that includes the safety system response to tool use or modified configuration of the safety system.

Type III – Development Process Support Tools – Type III tools are used in an off-line manner to support various software development activities. They are considered off-line because they do not have a direct impact or interaction with the safety-related software development process. Type III tools cannot introduce errors into the safety-related software/logic. Type III tools are used to create and maintain documentation and configuration management of the development process. The output of a Type III tool is documentation or specific file or data sets for the safety system. Type III tools can also be used to maintain configuration control over Type I, II, IV, and V tools.

Type IV – In-Line V&V Tools – Type IV tools use software or logic input from the output of a Type I tool to perform or assist software or logic verification or validation activities. Type IV tools are considered in-line tools because the safety-related software or logic can be automatically or manually changed by the tool user based on the results of the V&V process. The output of a Type IV tool is adjusted or modified software or logic that has been corrected to meet the requirements programmed into the tool itself.

Type V – Off-Line V&V Tools – Type V tools use software or logic input from the output of a Type I tool to perform or assist software or logic verification or validation activities. However, they differ from Type IV tools in that they do not modify or manipulate the software or logic that is being tested. Type V tools produce test results which provide a tool user with information needed to validate that the software or logic is performing its required design functions. If the V&V activities identify the need for software or logic changes, then that information is provided as feedback to the software developer who uses a Type I tool to make the change.

An ISL observation, based on an analysis of these tool types, is that the five tool types do not cover all the tools discussed in Section 4 of this report or all the tools used in the entire software development process. ISL also observes that the use of the terms “in-line” and “off-line” are inconsistent with the use of the same terms in IEC 61508 which could lead to confusion. ISL recommends that the NRC consider influencing the IEEE update of IEEE 7-4.3.2 to use the nine tool types identified in Section 4 of this report which are based on the categorization of tools in IEEE 1462 and IEEE 14102.

2.4.1.2 *IEEE 603*

The following discussion and analysis primarily applies to IEEE 603-2009; however, the discussion also applies the IEEE 603-1991 which is currently endorsed by RG 1.153. IEEE 603 establishes minimum functional and design criteria for the power, instrumentation, and control systems of nuclear plant safety systems. The functional and design criteria promote appropriate practices for design and evaluation of safety system performance and reliability. The criteria do not necessarily apply to all SSCs required for plant safety. An analysis of the overall plant

response to postulated design basis events determines which systems are subject to these criteria.

IEEE 603 does not discuss software tools; however, IEEE 603 contains design criteria applicable to digital computers, programs, software, and firmware that should be considered for applicability to software tools. The first design criterion applicable to safety system equipment employing digital computers, programs, or firmware is that components shall be of a quality that is consistent with minimum failure requirements and low failure rates and designed in accordance with a prescribed QA program. An ISL observation, based on our analysis of this high-level criterion, is that it could be applied to software tools by requiring software tools to be of a quality consistent with low failure rates and designed in accordance with a QA program that meets 10 CFR Part 50, Appendix B. However, ISL considers that it would be inappropriate to subject all software tools to this criterion. For this criterion to be applicable to all software tools, the criterion would have to be modified to reference or implement a software tool classification system that could be used to select methods of determining the quality of tools, assessing failure rates, and determining what QA program activities are required for each class of tools. Without these modifications, this high-level criterion is not suitable for application to software tools.

The second design criterion applicable to safety system equipment employing digital computers, programs, or firmware is that safety system equipment shall be qualified by testing, previous operating experience, or analysis, or any combination of these three methods, to substantiate that the equipment meets its performance requirements. An ISL observation, based on our analysis of this high-level criterion, is that it could be applied to software tools by requiring tool qualification using any combination of testing, previous operating experience, or analysis. The analysis would have to evaluate whether the tool was developed in accordance with a high-quality software development life cycle process and how the software tool was verified and validated. Without clarifying the type of analysis required for software tools, this criterion is not suitable for application to software tools.

A third design criterion applicable to safety system equipment employing digital computer programs and software is that the identification of the computer software and programs shall be distinctly identified by specifying the version number. An ISL observation, based on our analysis of this requirement, is that it could apply to software tools used to develop the digital computer programs and software used in safety systems. However, this high-level criterion, if applied to software tools, would merely require identification of the versions of software tools and not the content or location of the information. Therefore, this high-level criterion would be inadequate for application to software tools.

A fourth design criterion applicable to safety systems for which either quantitative or qualitative reliability goals have been established, analysis of the design shall be performed to confirm that the goals have been achieved. An ISL observation, based on our analysis of this design criterion, is that although a reliability analysis is applicable to digital computers, programs, or firmware employed in safety systems, the reliability of software tools is not a critical characteristic for using the tool to develop safety system software.

A fifth design criterion applicable to software common-cause failures is that an engineering evaluation of software common-cause failures should be performed. The evaluation should consider possible manual action and actions by nonsafety-related systems and components to accomplish the function that would be defeated by the common-cause failure. IEEE 603 defines common-cause failure as the “loss of function to multiple structures, systems, or components due to a shared root cause.” An ISL observation, based on an analysis of this concept, is that common-cause failure should be extended to software tools. A common-cause failure of safety system software could occur if the same software tool, that contains a defect, is used in the development of software for multiple systems or used in multiple phases of development for a single system. Software tool guidance should state that the tool should be sufficiently verified and validated to ensure that a defect in the tool does not cause a fault that can result in a common-cause failure in the safety-related system software if a single software tool is used during several phases of the software development life cycle or a single software tool is used in the same development life cycle for software in multiple safety-related systems.

2.4.1.3 *IEEE 1012*

The following discussion and analysis primarily applies to IEEE 1012-2012; however, IEEE 1012-2004 is also discussed because IEEE 1012-2004 is the version currently endorsed by Regulatory Guide 1.168. Both documents contain requirements for software V&V. V&V life cycle requirements are specified for each of four SILs and the requirements vary depending on which of the four SILs are assigned to the system, subsystem, component, or element. V&V processes, activities, and tasks include the analysis, evaluation, review, inspection, assessment, and testing of products. The two standards also define the required content and desired format of the V&V plan. The terminology in IEEE 1012-2012 has been revised from the previous version, IEEE 1012-2004, for consistency with the software life cycle processes specified in IEEE 12207-2008 and the system life cycle processes specified in IEEE 15288-2008 and to expand the scope of V&V processes to systems and hardware.

The key concepts in IEEE 1012-2012 include the use of the four SILs to describe the importance of the system, software, and hardware; the definition of minimum V&V tasks required for each SIL; the variable level of intensity and rigor applied to V&V tasks based on the SIL; and the definition of specific minimum criteria for each V&V task. Sections 1 through 4 of IEEE 1012-2012 discuss V&V objectives, definitions, references, and life cycle processes. Software tools are not discussed in Sections 1 through 4.

Section 5 of IEEE 1012-2012 discusses the SIL determination. SILs quantify the complexity, risk, and safety level of the system based on its importance. SILs define the minimum V&V tasks, activities, rigor, and level of intensity. Rigor is the required number of formal techniques and recording procedures used in the V&V processes. Intensity is the scope of analysis involved in each technique and procedure. IEEE 1012-2012 was developed for a four-level integrity schema to define the minimum V&V requirements; however, the standard supports any integrity level schema with proper mapping. A V&V criticality analysis classifies the integrity level of every system or element in the system of interest. The integrity level is assigned recursively to the components of each system element and to the functions of each element

with the intent of segregating parts that warrant high-integrity V&V tasks from those parts that do not. The integrity level of an element or function in a system can also be lower than the integrity level of the system if the element or function does not warrant a high-integrity classification. The integrity level assigned to reused, COTS, and GOTS components shall be the same as the system into which the reused, COTS, or GOTS components are integrated. The V&V criticality analysis also classifies the integrity level of enabling systems including software tools. IEEE 1012-2012 states that tools that insert or translate code (e.g., optimizing compilers and auto-code generators) shall be assigned the same integrity level as the integrity level assigned to the software element that the tool affects. If the tool cannot be verified and validated, then the output of the tool shall be subject to the same level of V&V as the software element.

An ISL observation, based on our analysis of these requirements, is that IEEE 1012-2012 only requires software tools that generate code used in the final safety-related system software to be assigned the same integrity level and subject to the same V&V rigor and intensity as the software element affected by the tool. If the software tool V&V tasks cannot be performed, as might be the case for COTS tools, then the tool output should be assigned the same integrity level and require the same V&V rigor and intensity as the software element which contains the tool output. IEEE 1012-2012 does not address other tools; however, a previous version of the standard, IEEE 1012-1998, states that tools that do not insert or translate code shall be verified and validated to assure that the tool meets their operational requirements. ISL's overall observation is that IEEE 1012-2012 lacks important V&V requirements for tools that do not insert or translate code and that IEEE 1012-2012 should be expanded to include an integrity level determination and specification of the minimum V&V rigor and intensity for all types of tools.

Section 6 of IEEE 1012-2012 discusses an overview of the V&V processes covered by IEEE 1012-2012. This section describes the processes of IEEE 12207-2008 and IEEE 15288-2008 for which V&V activities and tasks are defined. This section does not discuss software tools.

Sections 7 through 10 of IEEE 1012-2012 discuss common, system, software, and hardware V&V activities, respectively. Configuration management V&V assesses the CM process to assure that a CM strategy is defined; configuration controls are in place; items requiring configuration controls are defined including tools integral to system development; and status of items under CM is available throughout the life cycle. V&V tasks supporting the common acquisition process include determining the extent of V&V necessary for tools that insert or translate code and estimating the V&V budget including test facilities and tools. V&V tasks supporting the system implementation and transition processes include verifying that the implementation strategy accounts for all tools needed to construct the system element and verifying that the transition strategy is comprehensive and includes identification of transition tools. V&V tasks supporting the software and hardware maintenance processes include a migration assessment as to whether the software requirements and implementation, and the hardware requirements and fabrication, address migration tools. An ISL observation, based on our analysis of these requirements, is that the extent of V&V should be expanded to all tools, not just tools that insert or translate code.

Sections 11 and 12 of IEEE 1012-2012 discuss V&V reporting requirements and the content of the V&V plan. IEEE 1012-2012 states that the V&V plan shall summarize the V&V resources, including staffing, facilities, tools, finances, and special procedural requirements. The plan shall also describe hardware and software V&V tools to be used in the V&V processes including acquisition, training, support, and qualification information for each tool.

IEEE 1012-2012, Annex C, provides guidance for independent V&V, stressing the importance of technical independence of tools. Independent V&V should use or develop independent test and analysis tools separate from the developer's tools. Sharing of computer support tools (e.g., compilers, assemblers, and utilities) is allowable if the independent V&V qualifies the tools to assure that the common tools do not contain errors that may mask errors in the system being analyzed and tested. Off-the-shelf tools with an extensive history of use do not need to be qualified but the input data for these tools should be verified. An ISL observation, based on our analysis of the Annex C guidance, is that history of use should not be solely relied upon to assure that off-the-shelf tools are suitable for use in independent V&V activities. History of use should be augmented with testing, source evaluation, or surveys of the tool provider development process.

2.4.1.4 *IEEE 14102 and 1462*

Both IEEE 14102-2010 and IEEE 1462-1998 are adoptions of ISO/IEC 14102. Since IEEE 1462-1998 is the older of the two, a majority of the discussion will be in reference to IEEE 14102-2010 except where IEEE 1462-1998 provides information valuable to the study. The stated purpose of these standards is to assist organizations in the proper evaluation and selection of computer aided software engineering (CASE) tools. For this study, CASE tools are synonymous with software tools. Even though these documents are considered standards, there are few requirements. This is because the documents describe a "best practices" methodology for selecting CASE tools rather than a requirements-based methodology.

As previously stated, these standards are designed to assist organizations in evaluating and selecting CASE tools. However, the sections describing the characteristics and sub-characteristics of the various tool types can be shaped into guidance for evaluation of tools that have already been selected. Furthermore, by recognizing these standards as an acceptable means of selecting reliable, safe tools, tool planners and users can be directed toward an easily evaluated set of documents describing the software tool selection process, relieving reviewers of much of the burden of sorting through non-standard documentation of the tools used.

2.4.2 **Summary**

IEEE standards discuss software tools as an alternative to manual means to prepare specific documentation and perform V&V activities. IEEE standards also discuss concerns with common-cause failures. IEEE standards are currently endorsed by regulatory guides; however, regulatory guidance for the commercial nuclear power industry could be improved by endorsing the latest standards.

Three sets of IEEE standards specifically address software tools. The first set establishes criteria for digital computers in safety systems, IEEE 7-4.3.2-2003 and IEEE 7-4.3.2-2010. The

second set establishes the standards for V&V, IEEE 1012-2004, and IEEE 1012-2012. The third set establishes the IT standards for the evaluation and selection of CASE tools, IEEE 14102-2010 and IEEE 1462-1998. Only one of these sets of documents contains any specific guidance on evaluating the safety significance of a software tool. IEEE 7-4.3.2-2010 does not distinguish between the possible effects of using different types of tools and specifies the same requirements for all tools used in a development effort. However, the standard endorses qualifying a tool when it is developed within a high-quality life cycle process or by commercial dedication. IEEE 1012-2012 does define software tool integrity levels and the corresponding minimum V&V requirements; however, the IEEE 1012-2012 V&V requirements only apply to tools that insert or translate code and IEEE 7-4.3.2-2010 requires that all tools used in the development of safety-related systems meet the minimum V&V requirements of the highest safety integrity level.

IEEE 14102-2010 and IEEE 1462-1998 give guidance on how to evaluate and select appropriate and reliable tools but does not address any method for qualifying them for use in developing safety critical software. IEEE 603-2009 gives guidance on designing safety systems to address common-cause failures. Although the standard does not specifically address software tools, a defect in a software tool used to develop software in multiple systems or used in multiple life cycle phases to develop software for a safety system has the potential of introducing a fault which could result in a common-cause failure. Tools should be sufficiently verified and validated when used in this manner to prevent the possibility of a common-cause failure.

Based on the analysis of IEEE standards, the following essential elements of the documents relative to software tools have been determined.

- (1) IEEE 7-4.3.2-2003 and 7-4.3.2-2010 do not address the justification for the selection of software tools and acceptance criteria for compilers, operating systems, and libraries.
- (2) The proposed draft version of IEEE 7-4.3.2 requires that the benefits and risks associated with selecting software tools be taken into consideration and that the selected tools should limit the opportunity for making errors and introducing faults while maximizing the opportunity for detecting faults in the resulting safety-related PDD.
- (3) IEEE 7-4.3.2-2003, 7-4.3.2-2010, and the proposed draft require that software tools be maintained under configuration control and state that validation of a tool is not necessary if the tool is 100% tested.
- (4) IEEE 7-4.3.2-2003, IEEE 7-4.3.2-2010, and the proposed draft version require that software V&V satisfy the requirements for the highest safety integrity level (i.e., SIL 4) as defined in IEEE 1012-1998, IEEE 1012-2004, and IEEE 1012-2012, respectively.

- (5) IEEE 7-4.3.2-2003 requires that a tool validation test program be developed to provide confidence that the tool functions as required.
- (6) IEEE 7-4.3.2-2003 and the proposed draft version require that a software tool be used in a manner such that defects not detected by the tool will be detected by V&V activities.
- (7) IEEE 7-4.3.2-2010 requires that the output of the tool be subjected to the same level of V&V as the safety-related software being produced or that the tool be developed using the same or equivalent high-quality life cycle process as required for the produced software or be commercially dedicated.
- (8) The proposed draft version of IEEE 7-4.3.2 requires that a tool be developed using a quality life cycle process commensurate with the criticality of the safety functions performed by the end product.
- (9) The proposed draft version of IEEE 7-4.3.2 defines five tool categories: software development tools, surveillance and maintenance tools, development process support tools, in-line V&V tools, and off-line V&V tools.
- (10) Four SILs are used to describe the importance of the system, software, and hardware; the minimum V&V tasks required; the variable level of intensity and rigor applied to V&V tasks; and specific minimum criteria for each V&V task.
- (11) Tools that insert or translate code (e.g., optimizing compilers and auto-code generators) shall be assigned the same integrity level as the integrity level assigned to the software element that the tool affects.
- (12) If a tool cannot be verified and validated as might be the case for COTS tools, then the output of the tool shall be subject to the same level of V&V as the software element.
- (13) Independent V&V should use or develop independent test and analysis tools separate from the developer's tools.
- (14) Off-the-shelf tools with an extensive history of use do not need to be qualified but the input data for these tools should be verified.

Based on the analysis of IEEE standards, ISL has made the following important observations and recommendations.

- (1) An expectation that V&V activities will detect all defects not detected by a tool is not realistic and difficult to confirm.
- (2) The high-quality life cycle process development option in IEEE 7-4.3.2-2010 and the proposed draft version of IEEE 7-4.3.2 is not usually viable for COTS tools since COTS tools are typically developed for widespread use without regard for their application (i.e., nuclear safety-related software development).

- (3) IEEE 7-4.3.2-2010 offers an option of commercially dedicating the COTS tool, whereas the proposed draft version of IEEE 7-4.3.2 does not offer this option.
- (4) The five tool types defined in the draft version of IEEE 7-4.3.2 do not cover all CASE tools identified in IEEE 1462 and IEEE 14102. The NRC should consider influencing the proposed draft version of IEEE 7-4.3.2 to change the tool types to cover all CASE tools.
- (5) The terms “in-line” and “off-line” used in the draft version of IEEE 7-4.3.2 are inconsistent with the use of the same terms in IEC 61508.
- (6) Software tool guidance should state that the tool should be sufficiently verified and validated to ensure that a defect in the tool does not cause a fault that can result in a common-cause failure in the safety-related system software if a single software tool is used during several phases of the software development life cycle or a single software tool is used in the same development life cycle for software in multiple safety-related systems.
- (7) The use of four SILs defined in IEEE 2012 for safety-related software is not endorsed by NRC regulatory guides.
- (8) IEEE 1012-2012 lacks important V&V requirements for tools that do not insert or translate code; however, IEEE 1012-1998 states that tools that do not insert or translate code shall be verified and validated to assure that the tool meets their operational requirements.
- (9) The NRC should consider influencing the IEEE to expand the requirements of IEEE 1012-2012 to include an integrity level determination and specification of the minimum V&V rigor and intensity for all types of tools.
- (10) History of use should not be solely relied upon to assure that off-the-shelf tools are suitable for use in independent V&V activities. History of use should be augmented with testing, source evaluation, or surveys of the tool provider development process.

2.5 International Electrotechnical Commission Analysis

The International Electrotechnical Commission (IEC) is a worldwide organization for standardization with an objective of promoting international co-operation on all aspects of standardization in the electrical and electronic fields. This objective is met by publishing international standards, technical specifications, technical reports, publicly available specifications, and guides.

The IEC has a relatively long history of concern with software tools beginning in the 1980s with international standard IEC 880. IEC 880 interpreted the basic safety principles in hardwired systems for the utilization of digital systems in the safety systems of nuclear power plants. The IEC recognized at this time that the quality of the software tools used to develop safety system

software could have a direct effect on the quality of the software produced. In 2000, the IEC released IEC 60880-2, which contained extensive software tool requirements for use and ensuring quality. This was followed in 2006 by the second edition of IEC 60880, which incorporated and expanded software and software tool requirements for software used in computer-based systems performing category A functions. IEC 60880 is part of a series of standards that are intended to be used together.

IEC 61508 is an international standard that introduces the concept of functional safety, provides a risk-based approach for determining the required performance of safety systems, and provides generic requirements that can be used directly by industry or can be used as a basis for developing application-specific sector standards. IEC 61513 is a standard that provides the interpretation of the generic requirements of IEC 61508-1, IEC 61508-2, and IEC 61508-4 for I&C systems and equipment used to perform functions important to safety in the nuclear application sector. IEC 61513 refers directly to other standards for general topics related to categorization of functions and classification of systems (IEC 61226), coping with common cause failure (CCF) (IEC 62340), software aspects of computer-based systems (IEC 60880 and 62138), and hardware aspects of computer-based systems (IEC 60987). IEC 61513 and IEC 61226 together define the safety classes of I&C systems important to safety and the safety categories of I&C functions, respectively. Safety class 1 systems perform primarily category A safety functions but can perform category B, C, or nonsafety functions. Safety class 2 systems perform primarily category B safety functions but can perform category C or nonsafety functions. Safety class 3 systems perform primarily category C safety functions but can perform nonsafety functions. IEC 60880 provides the interpretation of the general requirements of IEC 61508-3 for software performing category A functions in the nuclear application sector. IEC 62138 provides the interpretation of the general requirements of IEC 61508-3 for software performing category B and C functions in the nuclear application sector.

2.5.1 Document Analysis

The following paragraphs discuss an analysis of four IEC standards that contain software tool requirements, guidance, and system classifications important to software tool requirements. These four standards are IEC 61508, 61226, 60880, and 62138. IEC 61508 introduces the concept of functional safety and provides generic requirements for software and software tools. IEC 61226 establishes a method for classifying I&C systems and equipment according to their safety significance. IEC 60880 and 62138 contain specific requirements and guidance for the evaluation and use of software tools. Both standards advocate the use of software tools in the development process to improve the productivity of the developers and to improve the quality of the software being developed. The requirements concerning the use of software tools include direction of how to document the use of the tools in the standard documentation required for software development. This includes the project planning documentation, the project QA documentation, and the software requirements specification. Additional requirements cover V&V of software tools and their outputs, integration testing, management of software written using application-oriented languages, and modification, replacement or upgrading software tools.

2.5.1.1 IEC 61508

IEC 61508 contains a total of 8 parts. Part 0 is a technical report that discusses functional safety. Parts 1 through 3 specify general, system, and software requirements for electrical, electronic, and programmable electronic (i.e., computer-based) safety-related systems. Part 4 contains the definitions and explanation of terms used throughout the IEC 61508 series of standards. Parts 5 through 7 provide guidance and examples for meeting the requirements of Parts 1 through 4.

2.5.1.1.1 IEC 61508-0

IEC 61508-0 introduces the concept of functional safety by first defining a few key terms used throughout the IEC 61508 international standard. IEC 61508-0 defines safety as a freedom from unacceptable risk of physical injury or of damage to the health of people, either directly, or indirectly, as a result of damage to property or to the environment. IEC 61508-0 defines functional safety as that part of safety that depends on a system or equipment operating correctly in response to its inputs. IEC 61508-0 defines safety-related as a term used to describe systems that are required to perform a specific function to ensure risks are kept at an acceptable level. The functions performed by safety-related systems are defined as safety functions and any system which carries out safety functions are safety-related systems.

An example of safety, given in the report, is insulation of high-temperature piping because the insulation protects against human injury. Insulation is not functional safety because the insulation is not an electrical, electronic, or computer-based component. An example of functional safety, given in the report, is a high-temperature trip of a motor using a thermal sensor. An ISL observation, based on our analysis of the example, is that the high-temperature trip doesn't fit the definition of functional safety because de-energizing an overheating motor protects the component and doesn't prevent physical injury or prevent damage to the health of people. The example of functional safety using a high-temperature trip could be improved if the motor can be tied to some hazardous physical situation for humans if the motor continued to overheat (e.g., starting a fire).

IEC 61508-0 states that a hazard analysis is necessary to identify significant hazards for equipment and any associated control system. The hazard analysis determines whether functional safety is necessary to ensure adequate protection against each hazard. IEC 61508-0 states that functional safety is only one method of dealing with hazards and is secondary to other means, including elimination or reduction of hazards through design.

IEC 61508-0 states that two types of requirements are necessary to achieve functional safety and both types of requirements are contained in IEC 61508. The first type of requirement is the safety function requirements that are derived from a hazard analysis. The second type of requirement is the safety integrity requirements that are derived from a risk assessment. The hazard analysis identifies what has to be done to avoid the hazardous event and the risk assessment gives the SIL required of the safety function for the risk to be acceptable.

IEC 61508-0 introduces SILs for specifying the target level of safety integrity for the safety functions to be implemented by the safety-related systems. IEC 61508-0 identifies four SILs (i.e., SIL1 through SIL4) but doesn't define them or provide guidance on how to determine the SIL for a safety function. Instead, the standard provides a risk-based conceptual framework and example techniques and states that a higher SIL necessitates greater rigor in the engineering of the safety-related system. An ISL observation, based on our analysis of the SILs, is that a risk assessment to determine the SIL of computer-based systems could be problematic since the frequency and type of failure of computer-based safety-related systems are not quantifiable. IEC 61508-0 supports this observation by discussing challenges to achieving functional safety for computer-based systems. IEC 61508-0 states that safety functions are increasingly being performed by electrical, electronic, or computer-based systems and that it is impossible in practice to fully determine every failure mode or to test all possible behaviors. IEC 61508-0 states that the challenge to achieving functional safety is to design the system to prevent failures or to control the failures when they arise.

IEC 61508-0 presents an example of developing functional safety requirements for a simple system containing a rotating blade protected by a hinged cover. The blade is accessed for routine cleaning by lifting the hinged cover which is interlocked so that whenever the cover is lifted an electrical circuit de-energizes the motor and applies a brake to stop the rotation of the blade. If the blade is not stopped, the operator could be injured. A hazard analysis identifies the safety function requirements that limit the maximum opening of the hinged cover before activating the interlock and how fast the interlock needs to be activated. A risk assessment determines the safety integrity requirements based on the probability of various sustainable injuries from a complete amputation of the hand to a bruise. The safety integrity requirements are also based on how frequently the cover is lifted for maintenance. The SIL required for this safety system increases with the severity of injury and the frequency of exposure to the hazard. Based on the safety function requirements and the safety integrity requirements, the functional safety requirements specification is developed. In this example, the functional safety requirements for the hazardous event are that the motor must be de-energized and the brake activated within 1 second when the hinged cover is lifted no more than 5 mm. The example merely states that the safety integrity level is SIL2 and does not discuss why an SIL2 has been assigned or how the risk-based assessment results in that classification. An ISL observation, based on an analysis of the example, is that the hazard analysis is deterministic and identifies all the possible hazards associated with cleaning the blade. This is analogous to evaluating the impact of software failures on systems which is also a deterministic process. The risk assessment, however, is based on probabilities of injury severity and frequency of accessing the hinged cover. The example does not provide any insight on how probabilities are assigned based on injury severity; however, the frequency of accessing the hinged cover is easily quantifiable based on maintenance requirements. These types of hardware-related probabilities are quantifiable as opposed to software failures which are not quantifiable and difficult to assess in a risk-based assessment.

The remainder of IEC 61508-0 discusses the content of IEC 61508 Parts 1 through 7 and identifies that IEC 61508 may be used as a basis for other industry standards, such as the railway, medical, and automotive industries, or that IEC 61508 can be used directly by industry.

IEC 61508 Parts 1 through 3 provide general, system, and software requirements, respectively, using an overall safety life cycle model from initial concept, through design and implementation, through operation and maintenance, to final decommissioning and disposal. Part 4 contains definitions and abbreviations including the definitions of SILs and tool classes. Parts 5, 6, and 7 do not contain requirements but provide guidance for meeting the requirements of Parts 1 through 4. Part 5 contains some examples of methods for the determining safety integrity levels and may shed some light on how SIL2 was determined for the example in this report. IEC 61508-0 does not make any references to software tools and is simply a high-level introduction to functional safety.

2.5.1.1.2 IEC 61508-1

IEC 61508-1 is Part 1 of the IEC 61508 series of standards and provides general requirements for the functional safety of electrical, electronic, and programmable electronic (computer-based) safety-related systems. Sections 1 through 4 of IEC 61508-1 contain high-level introductory material that describes the overall framework of the IEC 61508 series of standards, the applicability of the standard to safety-related systems, and how to conform to the standard. Sections 5 through 8 contain the requirements for the functional safety of safety-related systems and contain a few high-level requirements for tools. The tool requirements focus on adequate planning and documentation of tool use, especially the operating environment and tool versions.

Section 5 of IEC 61508-1 contains documentation requirements to ensure that all phases and associated activities of the system and software safety life cycles can be effectively performed. The documentation requirements emphasize the need for sufficient information to perform the required life cycle activities. Documentation should be concise, accurate, easy to understand, accessible, and maintainable. The only documentation requirement relevant to tools is that specific procedures may be necessary for managing multiple versions of documents produced by automatic or semi-automatic tools.

Section 6 of IEC 61508-1 contains requirements for the specification of responsibilities and activities of those who are responsible for the management of functional safety of a safety-related system. Management of functional safety is a life cycle support process that is performed in parallel with all other processes. Requirements for management of functional safety involve typical high-level management functions such as coordinating life cycle activities, identifying and assigning key personnel with appropriate competence, providing training, scheduling and managing audits, handling interface issues between various organizations, developing procedures that describe how to exchange information, how to track and resolve recommendations from the various life cycle activities, and how to implement CM of the safety-related systems. None of the requirements for the management of functional safety address tools or SILs.

Section 7 of IEC 61508-1 recommends an overall safety life cycle and contains requirements for each of the safety life cycle phases so that all activities necessary to achieve the required safety integrity of safety functions performed by safety-related systems can be dealt with in a systematic manner. The overall safety life cycle is a typical waterfall model and loosely follows the system and software life cycle processes defined in IEEE 12207-2008. The safety life cycle

phases contained in the IEC 61508 series of standards are tailored to implement safety-specific phases (i.e., hazard analysis and risk assessment).

The first safety life cycle phase is the concept phase with an objective of developing an understanding of the equipment under control (EUC) and its environment. Within this concept phase, the likely sources of hazards and their characteristics are determined. No requirements relevant to software tools are discussed in the concept phase.

The second safety life cycle phase is the overall scope definition phase with an objective of determining the boundaries of the EUC and the EUC control system and to specify the scope of the hazard and risk analyses. External events and types of initiating events are specified for the hazard and risk analyses. No requirements relevant to software tools are discussed in the overall scope definition phase.

The third safety life cycle phase is the hazard and risk analysis phase with an objective of determining the hazards, hazardous events, and hazardous situations relating to the EUC and the EUC control system, determining the event sequences leading to the hazardous events, and determining the risks associated with the hazardous events. The section requires that the hazard and risk analyses be completed and updated throughout the design process and states that a quantitative or qualitative technique can be used to meet the requirements. The requirement references IEC 61508-5 for guidance on performing the analyses. No requirements relevant to software tools are discussed in the hazard and risk analysis phase.

The fourth safety life cycle phase is the overall safety requirements phase with an objective of developing the specification for the overall safety requirements. The overall safety requirements are specified in terms of the overall safety function requirements and overall safety integrity requirements for the safety-related systems. This section also discusses the use of risk reduction measures to achieve the required functional safety. The overall safety function requirements is a set of all necessary safety functions required based on the hazardous events derived from the hazard analysis. The overall safety integrity requirements is a set of all necessary target safety integrity requirements that will result in the tolerable risk being met as determined by the risk analysis considering each overall safety function. The section provides guidance for determining which safety functions must be treated as safety-related and provides guidance for determining dangerous failure rates. No requirements relevant to software tools are discussed in the overall safety requirements phase.

The fifth safety life cycle phase is the overall safety requirements allocation phase with an objective of allocating the overall safety functions contained in both the overall safety function requirements and overall safety integrity requirements to the designated safety-related systems and other risk reduction measures. Another objective is to allocate a target failure measure and an associated SIL to each safety function executed by a safety-related system. The safety integrity requirements for each safety function are specified in terms of either the average probability of a dangerous failure for a low-demand mode of operation or the average frequency of a dangerous failure for a high-demand or a continuous mode of operation. A low-demand mode of operation is where the safety function is only performed on demand, to transfer the EUC into a specified safe state, where the frequency of demands is **no** greater than one per

year. A high-demand mode of operation is where the safety function is only performed on demand, to transfer the EUC into a specified safe state, where the frequency of demands is greater than one per year. A continuous mode of operation is where the safety function retains the EUC in a safe state as part of normal operation. Based on the average probability or frequency of a dangerous failure, the SIL for each safety function is determined from the following two tables.

Table 2.2 –Target Failure Measure for Low-Demand Mode of Operation

Safety Integrity Level (SIL)	Average Probability of a Dangerous Failure on Demand
4	$\geq 10^{-5}$ to $< 10^{-4}$
3	$\geq 10^{-4}$ to $< 10^{-3}$
2	$\geq 10^{-3}$ to $< 10^{-2}$
1	$\geq 10^{-2}$ to $< 10^{-1}$

Table 2.3 –Target Failure Measure for High-Demand or Continuous Mode of Operation

Safety Integrity Level (SIL)	Average Frequency of a Dangerous Failure [hr^{-1}]
4	$\geq 10^{-9}$ to $< 10^{-8}$
3	$\geq 10^{-8}$ to $< 10^{-7}$
2	$\geq 10^{-7}$ to $< 10^{-6}$
1	$\geq 10^{-6}$ to $< 10^{-5}$

In cases where the allocation process results in a safety-related system implementing an SIL4 safety function, risk parameters should be reviewed to determine if the SIL4 safety function can be avoided since the requirements for an SIL4 safety-related system are extensive. An ISL observation, based on an analysis of Tables 2.2 and 2.3 is that the ordering of the columns in the tables is a bit confusing and implies that the SIL determines the probability of failure. This is only true when a qualitative method is used to determine the SIL. In that case, the target failure measure is derived from the qualitatively determined SIL. For a quantitative method, the target failure measure is determined from available data and then a SIL is assigned based on that target failure measure. The SIL is not used in IEC 61508-1 as a parameter to determine the

qualification requirements of tools; however, the SIL is used to recommend, highly recommend, or not recommend the selection and use of tools for certain life cycle activities.

The sixth safety life cycle phase is the overall operation and maintenance planning phase with an objective of developing a plan for operating and maintaining safety-related systems to ensure that the required functional safety is maintained during operation and maintenance. The plan identifies routine maintenance, actions and constraints necessary to prevent an unsafe state or reduce the consequences of harmful events, and required documentation of operation and maintenance activities. No requirements relevant to software tools are discussed in the overall operation and maintenance planning phase.

The seventh safety life cycle phase is the overall safety validation planning phase with an objective of developing a plan for the overall safety validation of the safety-related system. The plan identifies relevant operating modes of the EUC operation and specifies the safety-related systems that need to be validated for each mode of EUC operation. The plan also identifies the technical strategy for validation and the measures, techniques, and procedures to be used for confirming that each safety function conforms to the overall safety functions requirements and the overall safety integrity requirements. The only requirement relevant to tools is that the validation plan shall include a discussion of the required environment in which the validation activities are to take place including calibrated tools and equipment for tests.

The eighth safety life cycle phase is the overall installation and commissioning planning phase with an objective of developing plans for the installation and commissioning of the safety-related system to ensure that the required functional safety is achieved. The plans identify the installation and commissioning schedule, responsible personnel, procedures, and criteria for declaring all or parts of the safety-related systems ready for installation. No requirements relevant to software tools are discussed in the overall installation and commissioning planning phase.

The ninth safety life cycle phase is the system safety requirements specification phase with an objective to define the system safety requirements in terms of system safety function requirements and system safety integrity requirements to achieve the required functional safety. The system safety requirements specification is derived from the allocation of safety requirements specified in the overall safety requirements allocation phase. The specifications contain safety function requirements and their associated SILs. No requirements relevant to software tools are discussed in the system safety requirements specification phase.

The tenth safety life cycle phase is the safety-related system realization phase with an objective of creating the safety-related systems conforming to the system safety requirements specifications. The requirements for creating the safety-related systems and for software tools are contained in IEC 61508-2 (systems) and IEC 61508-3 (software).

The eleventh safety life cycle phase is the specification and realization of other risk reduction measures phase. The objective of the other risk reduction measures specification and realization phase is to create other risk reduction measures to meet the safety functions requirements and safety integrity requirements specified for each system. Other risk reduction

measures are based on technology other than electrical, electronic, or programmable electronic (e.g., hydraulic, pneumatic) and are not discussed in IEC 61508-1. No requirements relevant to software tools are discussed in this phase.

The twelfth safety life cycle phase is the overall installation and commissioning phase. The objective of the overall installation and commissioning phase is to install and commission the safety-related systems as discussed in the installation plan. The requirements of this phase stress documentation of installation and commissioning activities and resolution of failures and incompatibilities. No requirements relevant to software tools are discussed in this phase.

The thirteenth safety life cycle phase is the overall safety validation phase with an objective of validating that the safety-related system meets the specification for the overall safety function requirements and overall safety integrity requirements. Validation activities must follow the overall safety validation plan and all equipment used for quantitative measurements as part of the validation activities must be calibrated. The overall safety validation phase requires that tools and equipment used, along with calibration data, be documented during validation.

The fourteenth safety life cycle phase is the overall operation, maintenance, and repair phase with an objective of ensuring that the functional safety of the safety-related system is maintained to the specified level and the technical requirements necessary for overall operation, maintenance, and repair are specified. This phase implements the plan for operating and maintaining the safety-related systems and implements the operation, maintenance, and repair procedures for the safety-related systems. Chronological documentation of operation, repair, and maintenance shall be maintained including the cause of demands on the safety-related system together with the performance of the system. No requirements relevant to software tools are discussed in the overall operation, maintenance, and repair phase.

The fifteenth safety life cycle phase is the overall modification and retrofit phase with an objective of defining the procedures that are necessary to ensure that the functional safety of the system is appropriate during and after modifications and retrofit. Modification and retrofit are initiated only through an authorized request detailing the hazards that may be affected, the proposed change, and the reason for the change. An impact analysis is required to assess the impact of the proposed modification or retrofit activity on the functional safety of the safety-related system. The impact analysis must include a hazard and risk analysis. Modifications or retrofits impacting functional safety trigger a return to the appropriate phase of the overall system or software safety life cycle phase and completion of all subsequent phases. No requirements relevant to software tools are discussed in the overall modification and retrofit phase.

The sixteenth safety life cycle phase is the decommissioning or disposal phase with an objective of defining the procedures necessary to ensure that the functional safety of the safety-related system is appropriate during and after the activities of decommissioning or disposing of the EUC. Prior to decommissioning or disposal of the EUC, an impact analysis, including a hazard and risk analysis, is required to assess the impact of the decommissioning or disposal activities on other EUCs and of any safety-related system associated with the EUC. Decommissioning or disposal activities impacting functional safety trigger a return to the appropriate phase of the

overall system or software safety life cycle phase and completion of all subsequent phases. No requirements relevant to software tools are discussed in the decommissioning and disposal phase.

The seventeenth safety life cycle phase is the verification phase with an objective of demonstrating that the outputs of each phase of the overall system and software safety life cycle meet all objectives and requirements specified for the phase. The verification phase requires that a verification plan be established and document tools to be used in the verification activities.

The functional safety assessment is the last process in the safety life cycle with an objective of specifying the activities necessary to determine the adequacy of the functional safety achieved by the safety-related system based on compliance with the requirements of the IEC 61508 series of standards. The functional safety assessment is applied to all life cycle phases and considers all the activities performed during each phase to judge whether adequate functional safety has been achieved. The requirements for the functional safety assessment stress the importance of independence of those performing the assessment. The minimum level of independence is specified as function of both the consequence that would arise in the event of a failure of all the risk reduction measures and the SIL. The functional safety assessment includes a discussion of the procedures, methods, and tools used for assessing the systematic capability and the hardware safety integrity to facilitate the re-use of the assessment results.

An ISL observation, based on our analysis of IEC 61508-1, is that IEC 61508-1 only contains high-level planning and documentation requirements for software tools that are necessary to track tool usage but do not address verification, validation, or qualification requirements of the tool itself. In summary, IEC 61508-1 provides general requirements for the functional safety of computer-based safety-related systems and recommends a safety-based life cycle process for developing safety-related systems. IEC 61508-1 identifies that specific procedures may be necessary for managing multiple versions of documents produced by automatic or semi-automatic tools. IEC 61508-1 requires adequate planning and documentation of tool use, especially the operating environment and tool versions. IEC 61508-1 assigns a SIL to the safety functions but the actual level is not used to determine the rigor of tool verification, validation, or qualification. IEC 61508-1 requires that the validation and verification plans include a discussion of the required environment in which the validation and verification activities are to take place including calibrated tools and equipment for tests. IEC 61508-1 also requires that the validation and verification phases document the tools and equipment used, along with calibration data.

2.5.1.1.3 IEC 61508-2

IEC 61508-2 is Part 2 of the IEC 61508 series of standards and provides system safety life cycle requirements and activities that are applied during the design and development of the safety-related systems. The system safety life cycle phases for the design and development of the safety-related system follow a typical waterfall life cycle model and loosely follow the system life cycle processes of IEEE 12207-2008. IEC 61508-2 applies to any safety-related system that contains at least one electrical, electronic, or programmable electronic element and applies

to all elements of that system including sensors, actuators, and the operator interface. IEC 61508-2 specifies how to refine the system safety requirements specification, developed in accordance with IEC 61508-1, into the system design requirements specification and specifies the requirements for activities that are to be applied during the design and manufacture of the safety-related system except for software that is covered in IEC 61508-3. Sections 1 through 6 of IEC 61508-2 contain high-level introductory material that describes the overall framework of the IEC 61508 series of standards and provides references and definitions. Section 7 contains electrical, electronic, and programmable electronic system safety life cycle objectives and requirements including a few high-level requirements for tools.

Section 7 of IEC 61508-2 contains all the requirements and objectives by life cycle phase. The system safety life cycle phases are similar to the life cycle phases discussed in IEC 61508-1. IEC 61508-2 describes the relationship between the system requirements of IEC 61508-2 and the software requirements of IEC 61508-3. The safety-related system design requirements specification and system architecture are developed in accordance with the requirements of IEC 61508-2. The hardware and software are designed concurrently in accordance with IEC 61508-2 and IEC 61508-3, respectively. After the hardware and software are designed, the system is integrated in accordance with the requirements of IEC 61508-2.

The first phase of the system safety life cycle is the system design requirements specification with an objective of specifying the design requirements for each safety-related system, in terms of subsystems and elements. The system design requirements specification is normally derived from the system safety requirements specification specified in IEC 61508-1 by decomposing the safety functions and allocating parts of the safety function to subsystems. Subsystems may be further decomposed into elements and architectures to satisfy the design and development requirements. The system design requirements specification contains details of all the hardware and software necessary to implement the required safety functions as specified by the system safety functions requirements specification and the details to achieve the SIL and required target failure measure for the safety function as specified by the system safety integrity requirements specification. No requirements relevant to software tools are discussed in the system design requirements specification phase.

The second phase of the system safety life cycle is the system safety validation planning phase with an objective to plan the validation of the safety of the safety-related system. Planning specifies the steps that are to be used to demonstrate that the safety-related system satisfies the system safety requirements specification from IEC 61508-1 and the system design requirements specification from phase one, discussed in the preceding paragraph. Planning for the validation considers the required environment in which the testing is to take place including all necessary tools and equipment and which tools and equipment should be calibrated.

The third phase of the system safety life cycle is the system design and development phase with an objective to design and develop the safety-related system, including ASICs if appropriate, to meet the system design requirements specification developed in phase one, discussed above. This third phase includes requirements for hardware safety integrity, special architectural requirements for integrated circuits with redundancy, requirements for systematic

safety integrity, requirements for system behavior on detection of a fault, and requirements for data communication processes. Where a safety-related system implements both safety and nonsafety functions, all hardware and software shall be treated as safety-related unless the implementation of the safety and nonsafety-related functions is sufficiently independent. The requirements for hardware and software shall be determined by the SIL of the safety function having the highest SIL unless the implementation of the safety functions is sufficiently independent. The documentation of the safety-related system design must specify and justify the techniques and measures to achieve the SIL. The design shall be based on a decomposition into subsystems with each subsystem having a specified design and set of integration tests. The design process should synthesize elements to achieve the required systematic capability through partitioning and independence and should consider the architectural constraints that effect hardware safety integrity. When determining the maximum SIL for specified architecture, the design process should consider hardware fault tolerance or component reliability data. The design and development phase shall quantify the effect of random hardware failures which shall be less than the target failure measure as specified in the system safety requirements specification. The design and development phase shall use appropriate techniques and measures to prevent introduction of faults in hardware and software and shall possess features for controlling systematic faults. The design and development phase shall specify the system behavior on detection of a fault. Only two requirements in this phase are relevant to tools. First, automatic testing tools and integrated development tools shall be used to avoid systematic faults and second, documentation of integration test planning shall include the test environment, tools, configuration, and programs.

The fourth phase of the system safety life cycle is the system integration phase with an objective to integrate and test the safety-related system. As part of the integration of all modules into the safety-related system, testing shall show that all modules interact correctly to perform their intended function and are designed not to perform unintended functions. This does not imply testing of all input combinations. Testing all equivalence classes may suffice and static analysis, dynamic analysis, or failure analysis may reduce the number of test cases to an acceptable level. The integration of software shall be carried out according to IEC 61508-3. System integration testing shall document the tools and equipment used along with calibration data.

The fifth phase of the system safety life cycle is the system operation and maintenance procedures phase with an objective to develop procedures to ensure that the required functional safety of the safety-related system is maintained during operation and maintenance. The operation and maintenance procedures specify the tools necessary for maintenance and revalidation and procedures for maintaining the tools and equipment.

The sixth phase of the system safety life cycle is the system safety validation phase with an objective to validate that the safety-related system meets in all respects the requirements for safety in terms of the required safety functions and safety integrity. Documentation of the validation testing shall state for each safety function the tools and equipment used, along with calibration data.

The seventh phase of the system safety life cycle is the system modification phase with an objective to make corrections, enhancements, or adaptations to the safety-related system, ensuring that the required safety integrity is achieved and maintained. Modifications shall be performed with at least the same level of expertise, automated tools, and planning and management as the initial development of the safety-related systems.

The eighth and last phase of the system safety life cycle is the system verification phase with an objective to test and evaluate the outputs of a given phase to ensure correctness and consistency with respect to the products and standards provided as input to that phase. The verification of safety-related systems shall be planned concurrently with the development, for each phase of the system safety life cycle, and shall be documented. Verification planning shall refer to all the criteria, techniques, and tools to be utilized in the verification for that phase.

IEC 61508-2 Annex B recommends techniques and measures, for each SIL, for satisfying IEC 61508-2 requirements and to avoid systematic failures during the different phases of the life cycle. Annex B, Tables B.1 through B.5, identify techniques and measures by SIL as either mandatory (M), highly recommended (HR), recommended (R), not recommended (NR), or no recommendation for or against the technique or measure is given (-). Tables B.1 through B.5 also identify the required effectiveness of the technique or measure (i.e., level of rigor) as either low, medium, or high. In summary, the SIL is used to select techniques and measures and determine the amount of rigor for satisfying the requirements of IEC 61508-2.

An ISL observation, based on our analysis of IEC 61508-2, is that IEC 61508-2 only contains high-level planning and documentation requirements for software tools that are necessary to track tool usage but do not address verification, validation, or qualification requirements of the tool itself. In summary, IEC 61508-2 requires that software tools are identified and discussed in plans and documentation of activities.

2.5.1.1.4 IEC 61508-3

IEC 61508-3 is Part 3 of the IEC 61508 series of standards and provides software requirements for safety life cycle phases and activities that are applied during the design and development of the safety-related software. The safety life cycle phases for the design and development of the safety-related software follow a typical waterfall life cycle model and loosely follow the software life cycle processes of IEEE 12207-2008.

IEC 61508-3 applies to any software forming part of a safety-related system or used to develop a safety-related system. Such software is termed safety-related software (including operating systems, system software, software in communication networks, human-computer interface functions, and firmware as well as application software). Sections 1 through 5 of IEC 61508-3 contain high-level introductory material that describes the overall framework of the IEC 61508 series of standards and provides references and definitions. Sections 6 and 7 of IEC 61508-3 discuss requirements for management of safety-related software and software safety life cycle requirements including requirements for support tools used to develop and configure a safety-related system such as development and design tools, language translators, testing and debugging tools, and CM tools.

Section 6 of IEC 61508-3 contains requirements for management of safety-related software that are in addition to the requirements specified in Section 6.2 of IEC 61508-1. The IEC 61508-3 additional requirements cover functional safety planning and SCM. One software tool relevant requirement of the IEC 61508-3 states that SCM shall maintain accurately and with unique identification all configuration items which are necessary to meet the safety integrity requirements of the safety-related system including all tools and development environments which are used to create or test, or carry out any action on, the software of the computer-based safety-related system.

Section 7 of IEC 61508-3 provides software safety life cycle requirements including some requirements for software tools. IEC 61508-3 provides a few general requirements applicable to the overall safety-related system development process. Section 7 of IEC 61508-3 also provides specific requirements for eight software safety life cycle processes including software safety requirements specification, validation planning, software design and development, computer-based hardware and software integration, software operation and modification procedures, software aspects of system safety validation, software modification, and software verification. Appropriate techniques and measures to satisfy the requirements for each life cycle phase are selected from a set of tables in Annexes A and B of IEC 61508-3 according to the SIL. Based on the SIL, certain techniques and measures are highly recommended, recommended, not recommended, or no recommendation for or against the technique or measure is made. The ranking of the techniques and measures is linked to the concept of effectiveness used in IEC 61508-2. Techniques that are highly recommended will be more effective in either preventing the introduction of systematic faults during software development, or more effective in controlling residual faults in the software revealed during execution than techniques which are just recommended. In summary, the SIL is used to determine which techniques and measures are most effective in controlling software faults while satisfying the IEC 61508-3 software development requirements; however, the SIL does not factor into the rigor associated with qualifying tools used during software development.

Section 7.1 of IEC 61508-3 contains general software safety life cycle requirements with an objective to structure the development of software into defined phases and activities. The requirements in this section are high-level requirements to select and specify a software life cycle model, divide the chosen software life cycle into phases and elementary activities, tailor the life cycle model consistent with the safety integrity and complexity of the project, integrate quality and safety assurance procedures into safety life cycle activities, use appropriate techniques and measures to execute the elementary activities of each life cycle phase, document the results of the activities in the software safety life cycle, and perform an impact analysis if at any phase of the software safety life cycle, a modification is required pertaining to an earlier life cycle phase. None of the general requirements pertain to software tools.

Section 7.2 of IEC 61508-3 contains requirements associated with the software safety requirements specification process with an objective to specify the requirements for the software safety functions for each safety-related system necessary to implement the required safety functions and to specify the requirements for software systematic capability for each safety-related system necessary to achieve the SIL specified for each safety function allocated to that

safety-related system. The requirements ensure that the software requirements specification is properly derived from the system safety requirements; sufficiently detailed to allow the design and implementation to achieve the required safety integrity; address independence through a CCF analysis; and address self-monitoring, self-testing, and fault detection during operation. The requirements apply to the safety-related software and not to software tools.

Section 7.3 of IEC 61508-3 contains requirements associated with establishing a validation plan for software aspects of system safety with an objective to develop a plan for validating the safety-related software aspects of system safety. The validation plan considers the details and technical strategy for the validation and the identification of the relevant modes of the EUC operation including reasonably foreseeable abnormal conditions and operator misuse. The validation plan also considers the required environment in which the validation activities are to take place including calibrated tools and equipment for testing.

Section 7.4 of IEC 61508-3 contains requirements for software design and development with an objective to design, implement, and verify software that fulfills the specified requirements for safety-related software with respect to the required SIL. An additional objective relevant to software tools is to select a suitable set of tools, including languages and compilers, run-time system interfaces, user interfaces, and data formats and representations for the required SIL, which assists verification, validation, assessment, and modification.

Section 7.4.2 of IEC 61508-3 contains general requirements for choosing a design method that stresses the incorporation of features to control complexity, enhance testability, support the capacity for modification and incorporate features for safe modification (e.g., modularity, information hiding, and encapsulation), and include features for self-monitoring of control flow and data flow for detecting and responding to failures.

Section 7.4.3 contains requirements for software architecture design that stress the need for a system or software architecture that uses architectural constraints to limit the impact of unsafe failures of an element and that development methods should take account of these constraints. The software architecture design shall be established by the software supplier or developer, or both, and shall select and justify an integrated set of techniques and measures necessary to satisfy the software safety requirements specification at the required SIL. The techniques and measures shall include design strategies for fault tolerance and avoidance including redundancy and diversity. Neither of these sections contain requirements for software tools.

Section 7.4.4 of IEC 61508-3 contains requirements for support tools. The selection of tools should consider the degree to which the tool supports the production of software with the required software properties, the clarity of the operation and functionality of the tool, and the correctness and repeatability of the output of the tool. The requirements for support tools cover both on-line support tools and off-line support tools. On-line support tools are software tools that can directly influence the safety-related system during its run time. The only requirement that applies to on-line tools is that on-line tools should be considered to be a software element of the safety-related system.

Off-line support tools are software tools that support a phase of the software development life cycle and cannot directly influence the safety-related system during its run time. Software off-line support tools are assigned a class of T1, T2, or T3 depending on whether the tool generates output which can directly or indirectly contribute to the executable code of the safety-related system or whether the tool can only fail to reveal defects in the executable software. IEC 61508-3 encourages the use of off-line support tools to increase the integrity of the software by reducing the likelihood of introducing or not detecting faults during the development. Requirements for off-line support tools stress the importance of justifying the selection of T1, T2, and T3 tools; having a specification or product documentation that clearly defines the behavior of T2 and T3 tools and any constraints on their use; determining the level of reliance placed on T2 and T3 tools and the potential failure mechanisms of the tools that may affect the executable software; and confirmation that T3 tools conform to their specification or documentation. Appropriate mitigation measures shall be implemented for all identified failure mechanisms in tool classes T2 and T3. Evidence that a class T3 tool conforms to its specification may be based on a combination of prior use and validation records. Finally, all off-line tools shall be validated and the validation results documented including a chronological record of the validation activities, the version of the tool product manual being used, the tool functions validated, tools and equipment used for the validation, the results of the validation activity including the pass or fail nature of the validation, test cases and their results, and discrepancies between expected and actual results.

Section 7.4.4 of IEC 61508-3 also contains a few additional requirements related to automatic code generation tools, CM of tools, and upgrading to new versions of tools. The suitability of automatic code generation tools for safety-related system development shall be assessed at the point in the development life cycle where development support tools are selected. Configuration management shall ensure that only qualified tools are used, only tools compatible with each other are used, and that information on tools that generate items in the configuration baseline is recorded in the configuration baseline including the identification of the tool and its version, the identification of the configuration baseline items for which the tool version has been used, and the way the tool was used including tool parameters, options and scripts selected. These CM requirements are to allow the baseline to be reconstructed if necessary. With respect to upgrading tools to new versions, IEC 61508-3 requires that each new version of off-line support tools be qualified. The qualification of the new version of the off-line support tool may rely on evidence provided for an earlier version provided that the functional differences will not affect tool compatibility with the rest of the toolset and the new version is unlikely to contain significant, new, unknown faults. Evidence that the new version is unlikely to contain significant new, unknown faults may be based on:

- (1) clear identification of the changes made,
- (2) an analysis of the V&V actions performed on the new version, and
- (3) any existing operational experience from other users that is relevant to the new version.

Section 7.4.5 of IEC 61508-3 contains requirements for detailed software system design involving the partitioning of the major elements in the architecture into a system of software modules; individual software module design; and coding. The section provides requirements to implement a structured design including organizing the software into a modular structure that separates out safety-related parts as far as possible, including range checking and other features that provide protection against data input mistakes, using previously verified software modules, and providing a design that facilitates future software modifications. This section contains no software tool requirements.

Section 7.4.6 of IEC 61508-3 contains requirements for code implementation. Source code shall be readable, understandable, and testable. Each module of software code shall be reviewed as part of the verification activity by inspection of the code by an individual, a software walk-through, or by a formal inspection. This section does not contain software tool requirements that may have generated the source code.

Section 7.4.7 of IEC 61508-3 contains requirements for software module testing. Each software module shall be verified to show whether it performs its intended function and does not perform unintended functions. If the module is simple, then an exhaustive test covering all possible inputs can be the most efficient way to demonstrate conformance. Otherwise, testing all equivalent classes or structure based testing may be sufficient and boundary value analysis or control flow analysis may reduce the test cases to an acceptable number. This section does not contain software tool requirements other than identifying tools used for software module testing.

Section 7.4.8 of IEC 61508-3 contains requirements for software integration testing. A software integration test specification is prepared during the design and development phase and includes the test cases, types of tests, test environment, tools, configuration, programs, test acceptance criteria, and procedures for corrective action on failure of a test. This section does not contain software tool requirements other than identifying tools used during testing.

Section 7.5 of IEC 61508-3 contains requirements for integrating hardware and software with an objective to integrate the software onto the target programmable electronic hardware and to combine the software and hardware in the safety-related programmable electronics to ensure their compatibility and to meet the requirements of the intended SIL. The objectives are met through the activities of specifying the integration tests, completing integration testing, and documenting the test cases and their results. The only requirement relevant to software tools is that the integration test specification shall describe the test environment including tools, support software, and configuration description.

Sections 7.6, 7.7, and 7.8 of IEC 61508-3 contain requirements for validation and software modification procedures. For each safety function, validation shall document the safety function being validated by test or analysis, the results of the validation activities, and the tools and equipment used together with the calibration data. Validation testing shall show that all specified requirements for safety-related software are correctly met and the software does not perform unintended functions. Software modifications shall be initiated only if authorized and supplemented with an impact analysis to determine whether a hazard and risk analysis is required and whether software safety life cycle phases need to be repeated. These three

sections contain no significant requirements relevant to software tools except for documenting the tools used.

Section 7.9 of IEC 61508-3 contains requirements for software verification with an objective to test and evaluate the outputs from all software phases to the extent required by the SIL and to ensure correctness and consistency with respect to the inputs to that phase. Section 7.9 covers verification activities in addition to software module verification testing, software integration verification testing, and programmable electronics verification testing required in previous life cycle phases. Software verification planning and documentation shall address the selection and utilization of verification tools.

An ISL observation, based on our analysis of IEC 61508-3, is that the IEC 61508-3 requirements for software tools are merely high-level planning, CM, and documentation requirements that are necessary to track tool usage but do not address the specific details of the verification, validation, or qualification of the software tool itself. IEC 61508-3 requires that software tools and development environments that are used to create or test, or carry out any action on, the software of the computer-based safety-related system are managed under a CM system, documented in a validation plan, and suitably selected for the required SIL. On-line software support tools are considered as part of the safety-related system. Off-line support tools are assigned a class of T1, T2, or T3 depending on the type of tool. Class T1 tools have the fewest requirements while class T3 tools have the most requirements. IEC 61508-3 requires all off-line support tools to be validated with appropriate documentation of the validation activities; however, IEC 61508-3 does not provide any details on the required rigor of validation activities that should depend on the class of software tool. ISL considers that the high-level software tool requirements in IEC 61508-3 are reasonable and acceptable because more detailed requirements are provided in lower-level documents IEC 60880 and IEC 62138. However, ISL observes that the tool class (i.e., T1, T2, and T3) is only used in IEC 61508-3 and should be carried through to lower-level documents IEC 60880 and IEC 62138 to customize the tool requirements based on the tool's importance in the design process.

2.5.1.1.5 IEC 61508-4

IEC 61508-4 is Part 4 of the IEC 61508 series of standards and contains the definitions and explanation of terms used throughout the IEC 61508 series of standards. Terms of particular interest are those that deal with the SIL and tools. These terms have been included in the glossary at the beginning of this document.

2.5.1.2 IEC 61226

IEC 61226 does not discuss the use, review, or approval of software tools nor does it establish requirements or provide guidance for software tools. However, IEC 61226 establishes a method of classification of the instrumentation and control (I&C) systems and equipment according to their importance to safety as required by IAEA NS-R-1, Requirement 5.1. Even though IAEA NS-R-1 has been superseded by IAEA SSR-2/1, the IEC 61226 classification methodology is consistent with the applicable requirements (i.e., IAEA NS-R-1, Requirement 5.1 and IAEA SSR-2/1, Requirement 22, of both documents. The classification of I&C functions

depends on their contribution to the prevention and mitigation of postulated initiating events. The resulting classification is then used in IEC 60880 and IEC 62138 to determine relevant specification and design requirements including qualification requirements for software tools used to develop I&C software and systems important to safety.

IEC 61226 is applicable to all the information and command functions and I&C systems and equipment that provide those functions. The functions, systems, and equipment under consideration provide automated protection, closed or open loop control, and information to the operating staff. The functions, systems, and equipment keep the nuclear power plant conditions inside the safe operating envelope and provide automatic actions, or enable manual actions, that prevent or mitigate accidents, or that prevent or minimize radioactive releases to the site or wider environment.

IAEA safety guides NS-G-1.3 and SSG-30 give guidance on the classification of systems according to the importance to safety of the functions they perform. IEC 61226 extends the IAEA NS-G-1.3 classification strategy and establishes the criteria and methods to be used to assign I&C functions of a nuclear power plant to one of the three categories (i.e., A, B, or C) depending on their importance to safety, or to an unclassified category for functions with no direct safety role. I&C functions falling within the boundary of the safety systems will generally be assigned to category A or B. I&C functions not within the boundary of a safety system but are defined as safety-related will generally be assigned to categories B or C. The descriptions of each category follow.

Category A: Functions that play a principal role in the achievement or maintenance of a non-hazardous stable state to prevent design basis events (DBE) from leading to unacceptable consequences. If specified manual actions are provided to reach the non-hazardous stable state, additional factors must be considered such as the availability of primary and alternate information sources and whether the manual actions are the only possibility for mitigation of this sequence of events. Category A also denotes functions whose failure could directly lead to accident conditions that may cause unacceptable consequences if not mitigated by other category A functions. Category A functions have high reliability requirements that may limit their functionality and complexity.

Category B: Functions that play a complementary role to the category A functions in the achievement or maintenance of a non-hazardous stable state to prevent a DBE from leading to unacceptable consequences or to mitigate or avoid equipment damage or activity release associated with a DBE. The operation of a category B function may avoid the need to initiate a category A function. Category B also denotes functions whose failure could initiate a DBE or worsen the severity of a DBE. Because of the presence of a category A function to provide the ultimate prevention of or mitigation of the consequences of a DBE, the safety requirements for the category B function need not be as high as those for the category A function. This allows, if necessary, the category B functions to be of higher functionality and complexity than category A functions.

Category C: Functions that play an auxiliary or indirect role in the achievement or maintenance of a non-hazardous stable state. Category C includes functions that have some safety significance, but are not category A or B. Category C functions can be part of the total response to a DBE but not be directly involved in mitigating the physical consequences of the accident, or be functions necessary for beyond design basis accidents.

IEC 61226 also provides detailed criteria for assigning functions to categories A, B and C. If a function does not meet any of the criteria, the function is "non-classified". The category of the function determines which IEC standard applies to software developed to satisfy that function. IEC 60880 provides requirements for software of computer-based systems performing category A safety functions. IEC 62138 provides requirements for software of computer-based systems performing category B and C safety functions. Both IEC 60880 and 62138 include requirements for software tools used to develop software that perform category A, B, and C functions.

2.5.1.3 IEC 60880

IEC 60880 Ed. 2.0 replaces the IEC 60880 Ed. 1.0 published in 1986 and IEC 60880-2 published in 2000. The new edition accounts for advancements in software engineering techniques; aligns the standard with new revisions of IAEA NS-R-1, which has been superseded by IAEA SSR 2/1, and NS-G-1.3; replaces requirements associated with standards published since the IEC 60880 Ed. 1.0 (e.g., IEC 61513, 61226, 62138, and 60987); and fully integrates IEC 60880-2 as chapters 13, 14, 15, and annexes G, H, and I. This standard addresses software aspects of safety class 1 computer-based systems performing category A functions. IEC 60880 is a nuclear power industry application of IEC 61508-3.

Section 1 of IEC 60880 briefly discusses the scope and objective of the standard. Sections 2 through 4 of IEC 60880 provide references, terms and definitions, and symbols and abbreviations. Sections 5 through 12 of IEC 60880 discuss software requirements associated with the various phases of the software life cycle from the software requirements specification to installation and operation. Specific requirements for tools are discussed in Section 14 of IEC 60880 with references to amplifying information in Section 15 of IEC 60880 pertaining to the qualification of pre-developed software.

Section 5 of IEC 60880 contains general requirements for software projects. The higher-level standard, IEC 61513, introduces the concept of the system safety life cycle as a method to control the system development processes. Section 5 of IEC 60880 extends the system safety life cycle to support the phased development of software. Section 5 of IEC 60880 acknowledges that some of the development phases can be automated by software tools and that the activities automated shall be documented including documentation of the inputs and outputs relevant to the phase. Section 5 of IEC 60880 also states that the QA plan shall require that tools used be identified and documented, and known to and mastered by persons involved in the software development process. With respect to tool identification, Section 5 of IEC 60880 also states that the QA plan shall require clear identification of which tools were used for each activity when several tools are used in the development process. The final requirement of Section 5 of IEC 60880 relevant to software tools states that CM shall identify all translation

tools and tool versions used to produce each executable entity. An ISL observation, based on our analysis of these IEC 60880 general requirements, is that the requirements are consistent with the high-level requirements of IEC 61508-3 which require identification, documentation, and CM of tools used in the software development process.

Section 6 of IEC 60880 contains requirements for specifying software requirements. Section 6 of IEC 60880 acknowledges that automated tools may be used to improve the coherence and completeness of the software requirements specification but does not contain any specific requirements for software tools used in that capacity.

Section 7 of IEC 60880 contains requirements for the design and implementation phase of software development. Section 7 of IEC 60880 encourages a tool-based approach for configuring pre-developed software to reduce the probability of human error and recommends the use of automated tools for language translation. The only IEC 60880, Section 7, requirements relevant to software tools are that tools for automated testing should be available and application-oriented languages should be transformed into a general purpose language by automated tools (e.g., by a code generator) prior to translation into an executable form. An ISL observation, based on our analysis of Section 7 of IEC 60880, is that the section does not contain requirements for the verification, validation, or qualification of tools used for automated testing and code generation. Another ISL observation is that automated testing tools are not classified and should be classified as class T2 tools consistent with IEC 61508 and should be verified, validated, or qualified as described in Section 14 of IEC 61508 since the tools can fail to detect errors in the safety-related software performing Category A functions. ISL further observes that code generation tools are not classified and should be classified as class T3 tools consistent with IEC 61508 and should be rigorously verified, validated, or qualified as described in Section 14 of IEC 61508 since the tools can introduce defects into the safety-related software performing Category A functions. Implementation of the T1, T2, and T3 tool classes from IEC 61508 in IEC 60880 would improve the consistency of the standards and allow tailoring of tool requirements based on class.

Section 8 of IEC 60880 contains requirements for software verification, stressing the need for independence from the developers. Section 8 of IEC 60880 contains several recommendations for tool use in the verification process and several requirements relevant to software tools used in the verification process. One requirement states that CASE tools with automatic code generation capability shall have a systematic structure to support effective verification. Another requirement states that a verification plan shall be established prior to starting software verification activities that documents all the tools to be utilized in the verification process and describes the activities to be performed to evaluate each tool involved in the software development process. Section 8 of IEC 60880 also requires that the verification team preparing the verification plan address the selection and utilization of software verification tools. The final Section 8 requirement relevant to software tools covers verification of the configuration of pre-developed software. If pre-developed software configuration data is produced using a tool-based approach that allows the data verification step to be omitted, the correctness of the tool shall be justified to provide sufficient confidence in the tool. An ISL observation, based on our analysis of Section 8 of IEC 60880, is that the section only addresses desired tool

characteristics; selection, planning, and documentation of tools; and planning and documentation of activities performed by tools. Section 8 of IEC 60880 also does not contain requirements for the verification, validation, or qualification of tools used for software verification. An ISL observation, based on an analysis of verification tools, is that verification tools should be classified as class T2 tools consistent with IEC 61508. Section 8 of IEC 60880 should address the verification, validation, or qualification of class T2 tools since the tools can fail to detect errors in safety-related software performing Category A functions. Implementation of the T1, T2, and T3 tool classes from IEC 61508 in IEC 60880 would improve the consistency of the standards and allow tailoring of tool requirements based on class.

Section 9 of IEC 60880 contains requirements related to the system integration of the hardware and software modules into subsystems and into the complete system. The only requirement relevant to software tools deals with integrated system verification. Section 9 of IEC 60880 requires that QA measures be established for software tools used for verification commensurate with the importance of the tools. Based on our analysis of Section 9 of IEC 60880, ISL agrees that the QA of software tools used for integrated system verification should be commensurate with the importance of the tools. Tailoring the QA of software tools based on their importance is consistent with regulations for software in 10 CFR Part 50, Appendix B. For consistency with IEC 61508, the tools in the integrated system verification toolset should be classified as a T1, T2, or T3 tool and should be verified, validated, or qualified as discussed in Section 14 of IEC 60880.

Section 10 of IEC 60880 contains requirements for testing to validate the system and its software. The only requirement relevant to software tools is that the validation report should identify all the software tools used in the validation process.

Section 11 of IEC 60880 contains requirements for modifying software that usually impacts both the executable code and the documentation. Software modifications may be requested during the development phase or after delivery. Section 11 of IEC 60880 requires the reassessment of tools if the tools will be used in the software modification and if the tools have been modified since the last time they were used.

Section 12 of IEC 60880 contains requirements for the interaction of operators with computer-based systems during installation and operation. Section 12 of IEC 60880 contains two security related requirements relevant to software tools. The first requirement is to identify the deviations from the normal plant security concepts necessary to perform planned software commissioning and modifications including the use of service stations and tools. The second requirement states that tools used for on-site software modifications should match the level appropriate to the tool's potential threat to the security of the system.

Section 13 of IEC 60880 contains requirements for establishing defenses against software design and coding faults that can lead to CCFs. The most important defense against CCFs is to produce software of the highest quality including self-monitoring features for data plausibility, parameter range checking, and loop timing. Section 13 of IEC 60880 does not contain any software tool related requirements; however, the section acknowledges that the use of well-developed software engineering methods with software tool support for software development

and verification can reduce the number of human design decisions and potentially reduce the number of software faults.

Section 14 of IEC 60880 expands the general IEC 60880 software requirements to cover software tools used in the development of software for computer-based safety systems of nuclear power plants. Section 14 of IEC 60880 begins with a brief overview of the use of software tools, their benefits, and which tools are within the scope of IEC 60880. Tools within the scope of IEC 60880 include tools that support the capture of requirements; tools used to transform requirements into system code and data; tools supporting verification, validation, and testing; tools used for the preparation and control of application data; and tools for the management and control of software development processes and products. IEC 60880 states that the requirements of IEC 60880 do not pertain to off-line tools used during the design and analysis of equipment important to safety such as word processors, project management tools, and other office administration tools not directly involved with software development. An ISL observation, based on our analysis of the use of the word “off-line” in IEC 60880, is that the use is slightly different than the definition of “off-line” tools in IEC 61508. IEC 61508 defines an “off-line” support tool as a software tool that supports a phase of the software development life cycle but cannot directly influence the safety-related system during its run time. IEC 60880 should state that the requirements of IEC 60880 do not pertain to class T1 off-line support tools such as word processors, project management tools, and other office administration tools. The use of tool classes in IEC 60880 would improve the consistency between IEC 61508 and IEC 60880 and allow the specification of tool requirements by class.

Section 14 of IEC 60880 also discusses the selection of tools focusing on tools that can directly or indirectly affect the quality of the software being produced. IEC 60880 requires that tools support the software engineering process; the limits of applicability of all tools be identified and documented; and tools not be used outside their limits of applicability without justification. IEC 60880 also requires that tools be verified and assessed to a level consistent with the tool reliability requirements, the type of tool, and the potential of the tool to introduce faults. Tools should also be sufficiently reliable so they do not jeopardize the end product. The reliability requirements of individual tools can be reduced by considering the principles of defense in depth and diversity.

Section 14 of IEC 60880 identifies the following types of tools which impact the required level of verification and assessment. An ISL observation, based on our analysis of these types of tools, is that the types should be assigned to tool classes of non-classified, T1, T2, or T3 for consistency with IEC 61508 so that the tool qualification requirements can be tailored to the different tool types.

- (1) Transformation tools (e.g., code generators, compilers)
- (2) V&V tools (e.g., static code analyzers, test coverage monitors, simulators)
- (3) Diagnostic tools (i.e., tools to maintain or monitor operating software)
- (4) Infrastructure tools (e.g., development support systems)

(5) Configuration control tools (e.g., version control tools)

IEC 60880, Section 14, addresses tool requirements by topic including the software engineering environment, tool qualification, tool CM, translators and compilers, application data tools, and test automation. Each of these topics are discussed in the following paragraphs.

Software Engineering Environment Requirements

IEC 60880 requires that an analysis of the software engineering environment and development processes be performed and documented to determine the strategy for providing tool support. If tools are not available, the development of new tools should be considered. The criteria for selection and evaluation of tools should be developed and prioritized to allow trade-offs prior to use. The modification, upgrade, or replacement strategy for tools shall be documented and justified. Tools used to provide diversity should be demonstrated to be dissimilar by showing that each tool was obtained from a different supplier, the tools have different input and output languages, or the tools have dissimilar requirements and design processes.

Tool Qualification Requirements

IEC 60880 requires that a tool qualification strategy be produced and the tool qualified using that strategy considering the reliability requirements of the tool and the type of tool. The reliability requirements of a tool shall be determined considering the consequences of a fault in the tool, the probability that a tool causes or induces faults in the software implementing the safety function, and what tools or processes mitigate the consequences of a fault in the tool. If tool output is included in the final software, the output should be systematically verified. Tools should be developed according to Sections 1 through 12 of IEC 60880 or evaluated and assessed according to the requirements in Section 15 of IEC 60880 unless the tool cannot introduce faults into the software, potential tool faults are mitigated, or the tool output is always systematically verified. The tool qualification strategy shall consider the following methods:

- (1) Analysis of tool development process and vendor tool history
- (2) Adequacy of tool documentation that supports verification of tool output and ease of learning
- (3) Testing or validation of the tool
- (4) Evaluation of prior tool use
- (5) Feedback of experience with tool use

An ISL observation, based on our analysis of the IEC 60880 software tool qualification requirements, is that the five methods identified are adequate to quality tools; however, IEC 60880 should specify that the primary method for qualifying

software tools should be testing or validation of the tool (Method 3) and analysis of the tool development process and vendor tool history (Method 1) with other methods providing a supporting role. Since IEC 60880 does not specify which methods are appropriate for which types of software tools, a tool could be qualified solely based on evaluation of prior use and feedback of experience from other users which is inadequate qualification for safety-related software tools. IEC 60880 should be more prescriptive and define which combinations of methods are required for each type of tool.

Tool Configuration Management Requirements

IEC 60880 requires that all tools and tool parameters used to generate base lined software be under CM. Any modification of a tool shall be verified and assessed. Records documenting the error history and limitations of the tool shall be maintained throughout the life of any tool whose output can introduce a fault into the final software.

Translator and Compiler Requirements

IEC 60880 acknowledges that the size and complexity of compilers can make it difficult to demonstrate that a compiler works correctly. Increased confidence can be gained through extensive experience using a compiler. The requirements for compilers and translators state that compilers and translators shall not remove defensive programming or error-checking features without warning and that compiler optimization should be avoided and shall not be used if it produces object code that is excessively difficult to understand, debug, test, and validate. If optimization is used, tests, verification, and validation shall be performed on the optimized code. IEC 60880 discourages the use of optimization unless absolutely necessary to meet performance requirements due to constraints in hardware speed and storage limits. In lieu of optimization, the design should consider assembly code and changing the hardware platform. The final requirements state that libraries shall be treated as pre-developed software that meet the requirements of Section 15 of IEC 60880 and that tests, verification, and validation shall be completed to ensure the correctness of any additional code introduced by the translator that is not directly traceable to the original source code. An ISL observation, based on our analysis of translator and compiler requirements, is that discouraging use of optimization is appropriate for safety-related tools because it makes V&V of the optimized code much more difficult; however, the recommended use of assembly code in lieu of optimization should be reconsidered since V&V and traceability of assembly code to the requirements specification is very difficult and requires specialized skills.

Application Data Tool Requirements

IEC 60880 requirements related to the preparation, verification, validation, and management of data for on-line use include proper documentation and definition of the design of the software system application data and its method of derivation from

the plant application data. Application data that can be changed during operation by the operator shall be identified along with the methods used to control changes of such parameters and shall not corrupt other data and code on the runtime system. Data should be in a form that supports verification, the procedures for verifying data shall be as formal as the procedures for verifying software, and an on-site facility shall be provided for the verification activities. Finally, data interfacing between two systems should be automatically generated from the same database and data that modifies or controls software functionality must be documented, justified, and tested.

Testing Automation Tools

IEC 60880 acknowledges that automation increases the amount of testing that can be performed in a given period and recommends the consideration of tools including test generators, test coverage analyzers, test drivers, on-line diagnostic programs, debuggers, and automated test suites. The requirements for automated testing tools state that the tools should record a complete test log, the tools should be appropriate to simulate the behavior of the executable code on the target system, and the tools should be appropriate to verify that the right executable is loaded correctly on the target system.

Section 15 of IEC 60880 contains requirements for the use of pre-developed software in computer-based systems and are part of the qualification requirements of the system in which pre-developed software is integrated. Section 15 of IEC 60880 does not contain requirements for software tools used to qualify pre-developed software.

IEC 60880, Annexes A through G, discuss the use of tools but add no additional tool requirements to those provided in Sections 1 through 15 of IEC 60880. Annex H of IEC 60880, in particular, provides an extensive discussion of tools for the production and checking of the requirements specification, design, and implementation but provides no guidance or requirements for those tools.

2.5.1.4 IEC 62138

IEC 62138 Ed. 1.0 provides software requirements for class 2 and 3 computer-based systems performing category B or C safety functions as defined by IEC 61226. IEC 62138 Ed. 1.0 complements IEC 60880 and IEC 60880-2 which provide software requirements for class 1 computer-based systems performing category A safety functions. By extension, IEC 62138 Ed. 1.0 complements IEC 60880 Ed. 2.0 which updates and consolidates the requirements of IEC 60880 and IEC 60880-2. IEC 62138 Ed. 1.0 also complements IEC 61513 which contains requirements that are not specific to software.

IEC 62138 organizes requirements by software development process in terms of a software safety life cycle. This life cycle is similar to the software development life cycle described in IEEE 12207-2008 with an emphasis on the stages that can affect software safety. Sections 1 through 3 contain a high-level discussion of the relationship of IEC 62138 with other standards, a discussion of safety classes of systems important to safety, and provides references and

definitions of key terms. Section 4 of IEC 62138 contains key concepts and assumptions for software development for I&C class 2 or 3 systems. Section 5 of IEC 62138 contains software requirements for class 3 systems performing category C functions. Section 6 of IEC 62138 contains software requirements for class 2 systems performing category B functions.

One key concept in Section 4 of IEC 62138 is that software tools are considered support system software that is part of the system software in a typical I&C system and is either off-line or on-line (i.e., embedded in nonsafety support systems). IEC 62138 acknowledges that system developers may specify system requirements in graphical form and that tools may automatically translate the graphical form into executable application software. Section 4 of IEC 62138 encourages the use of such tools if the tools are of adequate quality to reduce the risk of faults. The principles for safety class 2 systems contain more stringent requirements for selection and use of software tools than the principles for safety class 3 systems.

Section 5 of IEC 62138 includes requirements and guidance for software tools used in the development of software performing category C functions. IEC 62138 states that the QA plan must require that (1) tools that influence the correctness of software or system design be identified and documented; (2) tools are known by, and mastered by, the persons involved; and (3) user documentation for each tool be provided. IEC 62138 also requires that if several tools are used, the QA plan shall require clear identification of which tools are used for each activity.

IEC 62138 requires that verifications and reviews be performed according to documented provisions and that the extent of verification and review activities may be dependent on the tools used. IEC 62138 also requires that CM be applied to tools influencing the correctness of the software or system design and that the CM plan specify which tools are placed under CM.

Section 5 of IEC 62138 also contains guidelines for the selection and use of software tools for developing software that performs category C functions. These guidelines advise that the compatibility of a tool with other tools in use should be considered along with the tool quality. The guidelines also recommend that a well-known tool with extensive operating experience is preferable to an untried tool with little or no operating experience. Further guidance states that tools used should reduce the risk of introducing faults in new software, in the configuration of pre-developed software, and in the design of the system. The final software tool guidance states that evidence should be provided regarding the quality of the tool based on operational experience, tool certification, certification of the development practices of the tool suppliers, guarantee of appropriate tool development processes, or tests. An ISL observation, based on our analysis of the software tool requirements used in the development of software performing category C functions, is that the requirements are less rigorous but consistent with the IEC 60880 requirements for software performing category A functions. The software tool requirements include identification, documentation, and CM of tools. However, IEC 62138 only provides guidance for the selection, use, compatibility, and quality of software tools used to develop software performing category C functions, focusing on tool certification and certification of development practices rather than tool qualification required by IEC 60880 for tools used to develop software performing category A functions.

Section 6 of IEC 62138 contains the requirements for software performing category B functions. The requirements for software performing category B functions include all the requirements for software performing category C functions plus additional, more stringent requirements, commensurate with the higher safety importance of category B functions. One of the more stringent requirements states that the QA plan shall distinguish between tools which can introduce faults from those which overlook existing faults; thereby distinguishing between development tools and verification tools.

Other more stringent requirements in IEC 62138 for software performing category B functions is that tools that can introduce faults in software or system design shall be selected and used according to documented procedures and rules aimed at reducing the risk of introducing the fault. Also for these types of tools, evidence shall be provided regarding the tool quality and the tool's ability to produce the correct results and tool use shall be traced so that the tools used to generate an item or information may be identified. Evidence of tool quality may be based on operational experience, tool qualification or certification, certification of the development practices of the tool suppliers, guarantee of appropriate tool development processes, or tests. New tools or tool versions that have the potential of introducing faults in software or in system design shall also be assessed to ensure their compatibility with the previous tool or tool version. With respect to the selection of languages, an additional requirement for tools used to develop software performing category B functions states that general purpose languages used should have features facilitating tool supported static analyses of programs. An ISL observation, based on our analysis of these requirements, is that the tool classes defined in IEC 61508 (i.e., non-classified, T1, T2, and T3) should be used in IEC 62138 to improve the consistency in nomenclature and classification of tools in related IEC standards. As expected and as appropriate, based on an analysis of the software tool requirements used in the development of software performing category B functions, ISL observes that the requirements are more rigorous than the requirements for tools used to develop software performing category C functions but less rigorous than the requirements for tools used to develop software performing category A functions.

2.5.2 Summary

IEC 61508 describes a generic probabilistic risk-based process for safety-related software development and contains requirements and guidance for software tool selection, use, documentation, validation, and CM. IEC 60880 and IEC 62138 are commercial nuclear power specific applications of that generic process. IEC standards provide comprehensive, well-organized software tool qualification requirements that would form a sound technical basis for developing new regulatory guidance for the commercial nuclear power industry.

IEC 61508 introduces the concept of functional safety and the role of a hazard analysis and risk assessment. A hazard analysis generates safety function requirements that must be met to avoid hazardous events. A risk assessment generates safety integrity requirements and defines the SIL required of the safety function for the risk to be acceptable. IEC 61508 introduces SILs for specifying the target level of safety integrity for the safety functions to be implemented by the safety-related systems. IEC 61508 identifies four SILs (i.e., SIL1 through SIL4) and provides a risk-based conceptual framework and example techniques for determining the appropriate SIL.

IEC 61508 states that a higher SIL necessitates greater rigor in the engineering of the safety-related system. An ISL observation, based on our analysis of this risk-based methodology, is that a risk assessment to determine the SIL of computer-based systems could be problematic since the frequency and type of failure of computer-based safety-related systems are not quantifiable. This observation is supported by IEC 61508 which states that safety functions are increasingly being performed by electrical, electronic, or computer-based systems and that it is impossible in practice to fully determine every failure mode or to test all possible behaviors.

IEC 61508 requires that the use of tools be planned and well documented, especially the operating environment and tool versions. IEC 61508 states that specific procedures may be necessary to manage multiple versions of documents automatically produced by software tools. IEC 61508 also requires that software tools be discussed in V&V plans and documented during the V&V activities. IEC 61508 requires that SCM maintains an accurate and unique identification of all configuration items including tools and development environments.

IEC 61508 provides guidance for the selection of tools and requirements for justifying the selection of tools, having documentation that describes the behavior of the tool, determining the reliance on tools, and determining the potential failure mechanisms of tools. On-line software tools that can directly influence the safety-related system during its run time must be treated as an element of the safety-related system. Off-line support tools that support a phase of the software development life cycle and cannot directly influence the safety-related system during its run time are assigned to a class of T1, T2, or T3 depending on the tool's function. The selection, documentation, and validation requirements for tools are more rigorous for higher classes of tools (i.e., T3); however, the requirements are high-level and the concept of tool classes is not carried over to lower-level IEC standards (i.e., IEC 60880 and IEC 62138) to customize the tool qualification requirements based on the tool class.

IEC 61226 establishes a method of classification of I&C systems and equipment according to their importance to safety. The classification of I&C functions (i.e., A, B, C, or non-classified) depends on their contribution to the prevention and mitigation of postulated initiating events. The resulting classification is then used in IEC 60880 and IEC 62138 to determine relevant specification and design requirements including qualification requirements for software tools used to develop I&C software and systems important to safety.

IEC 60880 addresses software aspects of safety class 1 computer-based systems performing category A functions as defined by IEC 61226. IEC 60880 is a nuclear power industry application of IEC 61508-3. IEC 60880 requires that the QA plan identify and document all tools and that persons involved in the software development process must have knowledge and mastery of the tool. IEC 60880 requires all tools to be verified and assessed to a level consistent with tool reliability requirements, the type of tool, and the potential of the tool to introduce faults. IEC 60880 also identifies five types of tools that require different levels of verification and assessment. An ISL observation, based on our analysis of these five types of tools, is that the types should be assigned to tool classes of non-classified, T1, T2, or T3 for consistency with IEC 61508 and the tool verification and assessment (i.e., qualification) requirements should be tailored to the different tool classes rather than different tool types. IEC

60880 attempts to tailor the qualification requirements by requiring that tools be qualified considering the reliability requirements of the tool and the type of tool; however, IEC 60880 lacks specific details of the tailoring process. IEC 60880 states that the reliability requirements of a tool are determined considering the consequences of a fault in the tool, the probability that a tool causes or induces faults in the software implementing the safety function, and what tools or processes mitigate the consequences of a fault in the tool. IEC 60880 specifies five tool qualification methods but does not recommend any specific combination of methods for the different tool types. IEC 60880 does not require tool qualification if the tool cannot introduce faults into the software, the tool output is always systematically verified, or tool faults are mitigated. IEC 60880 requires that all tools and tool parameters be under CM and discourages the use of optimization when compiling the executable code.

IEC 62138 addresses software aspects of safety class 2 and 3 computer-based systems performing category B or C functions as defined by IEC 61226. IEC 62138 complements IEC 60880 and IEC 61513. IEC 62138 requirements for software tools are consistent with but less rigorous than the IEC 60880 requirements for software tools used to develop software performing category A functions. For software tools used to develop software performing category C functions, IEC 62138 focuses efforts on tool certification and the certification of the tool development process rather than a rigorous tool qualification process. For software tools used to develop software performing category B functions, IEC 62138 focuses efforts on tool certification or qualification, and the certification of the tool development process.

The IEC standards contain a generic safety-related software development process based on the use of hazard analysis to determine risks and a risk assessment to determine mitigation strategies. IEC standards also apply this generic process to the commercial nuclear power industry. Based on the analysis of IEC standards, the following essential elements of the documents relative to software tools have been determined.

- (1) A risk assessment is required to determine a SIL (i.e. SIL1 through SIL4) for each safety function in a safety-related system.
- (2) The safety integrity requirements for each safety function are specified in terms of either the average probability of a dangerous failure for a low-demand mode of operation or the average frequency of a dangerous failure for a high-demand or a continuous mode of operation.
- (3) Most safety life cycle phases require that tools and equipment used be documented during validation.
- (4) Safety functions are increasingly being performed by electrical, electronic, or computer-based systems and it is impossible in practice to fully determine every failure mode or to test all possible behaviors.
- (5) Where a safety-related system implements both safety and non-safety functions, all hardware and software must be treated as safety-related unless

- the implementation of the safety-related and nonsafety-related functions is sufficiently independent.
- (6) The selection of tools should consider the degree to which the tool supports the production of software with the required software properties, the clarity of the operation and functionality of the tool, and the correctness and repeatability of the output of the tool.
 - (7) On-line software support tools are considered as a software element of the safety-related system.
 - (8) Off-line software support tools are assigned a class of T1, T2, T3, or non-classified depending on whether the tool generates output which can directly or indirectly contribute to the executable code of the safety-related system or whether the tool can only fail to reveal defects in the executable software.
 - (9) The selection of T1, T2, and T3 tools must be justified.
 - (10) A specification or product documentation is required for T2 and T3 tools that clearly defines the behavior of the tools and any constraints on their use.
 - (11) The level of reliance placed on T2 and T3 tools and the potential failure mechanisms of the tools that may affect the executable software must be determined and appropriate mitigation measures implemented for all identified failure mechanisms.
 - (12) Confirmation that T3 tools conform to their specification or documentation is required based on a combination of prior use and validation records.
 - (13) All off-line tools shall be validated and the validation results documented.
 - (14) The suitability of automatic code generation tools for safety-related system development shall be assessed at the point in the development life cycle where development support tools are selected.
 - (15) Configuration management shall ensure that only qualified tools are used, only tools compatible with each other are used, and that information on tools that generate items in the configuration baseline is recorded in the configuration baseline.
 - (16) Each new version of off-line support tools must be qualified. Qualification may rely on evidence provided for an earlier version provided that the functional differences will not affect tool compatibility with the rest of the toolset and the new version is unlikely to contain significant, new, unknown faults.
 - (17) IEC 61226 assigns I&C functions to one of three categories (i.e., A, B, or C) according to their importance to safety of to an unclassified category for functions with no direct safety role. The category is then used in IEC 60880

and IEC 62138 to determine relevant specification and design requirements including qualification requirements for software tools.

- (18) IEC 60880 requires that tools be verified and assessed to a level consistent with the tool reliability requirements, the type of tool, and the potential of the tool to introduce faults.
- (19) The reliability requirements of a tool shall be determined considering the consequences of a fault in the tool, the probability that a tool causes or induces faults in the software implementing the safety function, and what tools or processes mitigate the consequences of a fault in the tool.
- (20) IEC 60880 identifies five types of tools: transformation tools, V&V tools, diagnostic tools, infrastructure tools, and configuration control tools.
- (21) If tool output is included in the final software, the output should be systematically verified.
- (22) Tools should be developed or evaluated and assessed unless the tool cannot introduce faults into the software, potential tool faults are mitigated, or the tool output is always systematically verified.
- (23) All tools must be qualified based on the tool reliability requirements and the type of tool based on: analysis of the tool development process and vendor history, adequacy of tool documentation, tool testing or validation, evaluation of prior tool use, or feedback or experience with tool use.
- (24) All tools and tool parameters used to generate base lined software are required to be under CM.
- (25) The use of optimization is discouraged unless absolutely necessary to meet performance requirements due to constraints in hardware speed and storage limits. In lieu of optimization, the design should consider assembly code and changing the hardware platform.

Based on the analysis of IEC standards, ISL has made the following important observations and recommendations.

- (1) The SIL is not used as a parameter to determine the qualification requirements of tools; however, the SIL is used to recommend, highly recommend, or not recommend the selection and use of tools for certain life cycle activities.
- (2) The concept of tool class (i.e., T1, T2, and T3) is only used in IEC 61508 and should be carried through to lower-level documents IEC 60880 and IEC 62138 to customize the tool requirements based on the tool's importance in the design process.

- (3) The types of tools identified in IEC 60880 should be assigned to tool classes of non-classified, T1, T2, or T3 for consistency with IEC 61508 so that the tool qualification requirements can be tailored to the different tool types
- (4) IEC 60880 does not specify which qualification methods are appropriate for which types of software tools; therefore, a tool could be qualified solely based on evaluation of prior use and feedback of experience from other users which is inadequate qualification for safety-related software tools.
- (5) The recommended use of assembly code in lieu of optimization should be reconsidered since V&V and traceability of assembly code to the requirements specification is very difficult and requires specialized skills.

2.6 Automotive Industry Analysis

Safety is one of the key issues of current and future automobile development. New functionalities such as collision avoidance and detection, blind spot monitoring, dynamic cruise control, drive-by-wire technology, propulsion, in-vehicle dynamics control, and active and passive safety systems increasingly touch the domain of system safety engineering. Development and integration of these functionalities require safe system development processes and the need to provide evidence that all reasonable system safety objectives are satisfied. With the trend of increasing technological complexity, software content, and mechatronic implementation, there are increasing risks from systematic failures and random hardware failures. Guidance to avoid these risks should be provided through appropriate requirements and processes.

2.6.1 Document Analysis

ISO 26262 is an international standard that is an automotive specific implementation of IEC 61508. ISO 26262 provides requirements for the functional safety of road vehicles. The following paragraphs discuss an analysis of ISO 26262 focusing on the use, selection, requirements, verification, validation, and qualification of software tools used in the automotive industry.

2.6.1.1 ISO 26262

ISO 26262 applies to all activities during the safety life cycle of safety-related systems that include one or more electrical or electronic systems installed in series production passenger cars. Safety is a key issue of future automobile development and new functionalities (e.g., driver assistance, propulsion, vehicle dynamic control, and active and passive safety systems) touch the domain of safety systems. Development and integration of these functionalities strengthen the need for safe system development processes and the need to provide evidence that all reasonable system safety objectives are satisfied. With the trend of increasing technological complexity and software content, there are increasing risks from systematic failures and random hardware failures. ISO 26262 includes guidance to avoid these risks by providing appropriate requirements and processes.

ISO 26262 provides an automotive safety life cycle and supports tailoring the necessary activities during those life cycle phases. ISO 26262 also provides an automotive-specific risk-based approach to determine automotive safety integrity levels (ASIL) and uses the ASIL to specify applicable requirements of ISO 26262 so as to avoid unreasonable residual risk. ISO 26262 provides requirements for validation and confirmation measures to ensure a sufficient and acceptable level of safety is achieved.

ISO 26262 consists of ten parts. ISO 26262-1 specifies the terms, definitions, and abbreviations for all parts of ISO 26262. ISO 26262-2 specifies the requirements for functional safety management for automotive applications including project-independent requirements for organizations involved and project-specific requirements for management activities in the safety life cycle. ISO 26262-3 specifies the requirements for the concept phase for automotive applications including item definition, initiation of the safety life cycle, hazard analysis and risk assessment, and functional safety concept. ISO 26262-4 specifies the requirements for product development at the system level for automotive applications including requirements for initiating product development at the system level, specification of technical safety requirements, the technical safety concept, system design, item integration and testing, safety validation, functional safety assessment, and product release. ISO 26262-5 specifies the requirements for product development at the hardware level for automotive applications including requirements for initiating product development at the hardware level, specification of hardware safety requirements, hardware design, hardware architectural metrics, evaluation of random hardware failures, and hardware integration and testing. ISO 26262-6 specifies the requirements for product development at the software level for automotive applications including requirements for initiating product development at the software level, specification of software safety requirements, software architectural design, software unit design and implementation, software unit testing, software integration and testing, and verification of software safety requirements. ISO 26262-7 specifies the requirements for production, operation, service, and decommissioning. ISO 26262-8 specifies the requirements for supporting processes including interfaces with distributed developments, overall management of safety requirements, CM, change management, verification, documentation, confidence in the use of software tools, qualification of software components, qualification of hardware components, and proven in use argument. ISO 26262-9 specifies the requirements for ASIL-oriented and safety-oriented analyses including tailoring requirements with respect to ASIL, criteria for coexistence of elements, analysis of dependent failures, and safety analyses. ISO 26262-10 provides an overview and additional explanations of ISO 26262 and is intended to enhance the understanding of the other parts of ISO 26262. ISO 26262-10 describes the general concepts of ISO 26262 to facilitate comprehension and expands the general concepts to specific contents. Only ISO 26262-6 and ISO 26262-8 contain relevant software tool requirements and guidance and are discussed in this report.

An ISL observation, based on our analysis of ISO 26262, is that there are three aspects of ISO 26262 that deviate from the IEC 61508 standard upon which ISO 26262 is based. The areas of difference are in the use of terms related to the SIL and in the manner in which methods are highly recommended, recommended, or not recommended for complying with the requirements in the standard. Even though ISO 26262 is an automotive-specific implementation of IEC

61508, ISO 26262 does not use the term SIL but instead uses ASIL. Also, the level of safety is defined with a letter from A (lowest) to D (highest) instead of with a number from 1 (lowest) to 4 (highest) as specified in IEC 61508. Finally, the IEC 61508 acronyms of M (mandatory), HR (highly recommended), R (recommended), NR (not recommended), and “-“ (no recommendation) are not used in ISO 26262. Instead, a highly recommended method is designated by the symbol “++”, recommended is designated with the symbol “+”, and no recommendation for or against the method is designated by the symbol “o”. ISO 26262 also does not make use of the required effectiveness of each method as a function of SIL as implemented in IEC 61508. These types of inconsistencies between standards should be avoided unless absolutely necessary to avoid unnecessary confusion. Since the NRC has no direct or indirect involvement in the development of ISO 26262 for the automotive industry, ISL does not recommend that the NRC attempt to influence changes in ISO 26262 to remove these inconsistencies. The inconsistencies between ISO 26262 and IEC 61508 are simply identified as a source of confusion that should be recognized and avoided when developing new regulatory guidance for commercial nuclear power.

2.6.1.1.1 ISO 26262-6

ISO 26262-6 specifies the requirements for product development at the software level for automotive applications. Sections 1 through 3 of ISO 26262-6 contain introductory material including references, terms, definitions, and acronyms. Section 4 contains general requirements for software development. Sections 1 through 4 do not discuss software tools.

Section 5 of ISO 26262-6 contains requirements for initiating product development at the software level. Specifically, the activities and the determination of appropriate methods for the product development at the software level shall be planned. The initiation of software development is a planning activity where the life cycle for software development is tailored to determine the sub-phases and planned according to the extent and complexity of the item development. The planning is initiated by determining the appropriate methods to comply with the requirements and their respective ASIL. The methods are supported by guidelines and tools, which are determined and planned for each sub-phase and supporting process. Planning of software development is coordinated with product development at the system and hardware levels.

Section 5 of ISO 26262-6 contains two requirements relevant to software tools. The first requirement is that tools used in the software development process shall be consistent across all the sub-phases of the tailored software life cycle and be compatible with the system and hardware development phases such that the required data can be transformed correctly. The second requirement is that for each sub-phase of software development, corresponding tools shall be selected including guidelines for their application. An ISL observation, based on our analysis of these requirements, is that the requirements are consistent with and enforce industry standard requirements for selecting tools that are well integrated and well documented. However, ISO 26262-6 should require an appropriate amount of training in the use of tools so that the guidelines for tool use can be safely implemented.

Section 5 of ISO 26262-6 also contains criteria that shall be considered when selecting a suitable modeling or programming language. The criteria include an unambiguous definition (e.g., syntax and semantics of the language); support for embedded real time software and runtime error handling; and support for modularity, abstraction, and structured constructs. Criteria not addressed by the language itself shall be covered by the corresponding guidelines, or by the development environment. Assembly languages can be used for those parts of the software where the use of high-level programming languages is not appropriate, such as low-level software with interfaces to hardware, interrupt handlers, or time-critical algorithms. An ISL observation, based on our analysis of the requirements for a programming language, is that the use of assembly language for programming low-level software would introduce several unique problems related to the verification, validation, and testing of the software. Bi-directional tracing of assembly language code to requirements specifications would be difficult. Assembly language code inspection is more difficult than code inspection for high-level programming languages and requires a more specialized set of skills which may inhibit QA activities and increase the probability of defects in the code. Verification of test case coverage would also be more difficult with code written in assembly language. Given these considerations, the use of assembly language to program safety-related software should be carefully scrutinized.

Section 5.4.7 of ISO 26262 discusses eight alternative guidelines for modeling and coding including enforcement of low complexity, use of language subsets, enforcement of strong typing, use of defensive implementation techniques, use of established design principles, use of unambiguous graphical representation, use of style guides, and use of naming conventions. An ISL observation, based on our analysis, is that the guidance for selecting topics to be covered by modeling and coding guidelines is vague. The instructions in ISO 26262 for interpreting the information in tables states that each row with the same number but a different letter represents an alternative method for completing the required activities. The instructions also state that an appropriate combination of alternative methods shall be applied in accordance with the ASIL indicated. The table in Section 5.4.7 of ISO 26262-6 for the highest ASIL (i.e., ASIL D) highly recommends all eight alternative topics. Based on the instructions provided, it is not clear whether all of the topics need to be covered for ASIL D safety-related modeling and coding guidelines or whether a subset of the eight topics is required.

The remaining sections of ISO 26262-6 contain requirements for subsequent phases of product development at the software level. Section 6 of ISO 26262-6 contains requirements for the specification of software safety requirements, including the hardware-software interface requirements, derived from the technical safety concept and the system design specification to support the subsequent design phases. Section 7 of ISO 26262-6 contains requirements for the software architectural design that realizes the software safety requirements specified in Section 6 of ISO 26262-6. Section 8 of ISO 26262-6 contains requirements for software unit design and implementation in accordance with the software architectural design defined in Section 7 of ISO 26262-6 and the associated software safety requirements specified in Section 6 of ISO 26262-6. Section 9 of ISO 26262-6 contains requirements for software unit testing to demonstrate that the software units fulfill the software unit design specifications and do not contain undesired functionality. Section 10 of ISO 26262-6 contains requirements for software integration and testing to demonstrate that the software architectural design is realized by the embedded

software. Section 11 of ISO 26262-6 contains requirements for the verification of software safety requirements to demonstrate that the embedded software fulfils the software safety requirements. None of these sections contain any requirements for software tools.

2.6.1.1.2 ISO 26262-8

ISO 26262-8 specifies the requirements for supporting processes including interfaces within distributed developments, overall management of safety requirements, CM, change management, verification, documentation, confidence in the use of software tools, qualification of software components, qualification of hardware components, and proven in use argument. Sections 1 through 3 of ISO 26262-8 contain introductory material including references, terms, definitions, and acronyms. Section 4 contains general requirements for software developments supporting processes. Sections 1 through 4 do not discuss software tools.

Section 5 of ISO 26262-8 contains requirements for the cooperative development of items and elements by the vehicle manufacturer and the supplier. This section only applies when the supplier shares some of the responsibility for the safety of a developed item. The selection of a supplier shall include an evaluation of the supplier's capability to develop and produce items and elements of comparable complexity and ASIL. The only requirement relevant to tools is that the vehicle manufacturer and the supplier shall specify a developer interface agreement that discusses the supporting processes and tools, including interfaces, to assure compatibility between vehicle manufacturer and supplier.

Section 6 of ISO 26262-8 contains requirements to ensure the correct specification of safety requirements and to ensure consistent management of safety requirements throughout the entire safety life cycle. Safety requirements constitute all requirements aimed at achieving and ensuring the required ASILs allocated or distributed among the elements. The management of safety requirements includes managing requirements, obtaining agreement on the requirements, obtaining commitments from those implementing the requirements, and maintaining traceability. The recommended method of specifying and verifying safety requirements varies by ASIL. The use of suitable requirements management tools is recommended although no specific requirements are included in Section 6 for their selection, verification, validation, or use.

Section 7 of ISO 26262-8 contains requirements for CM to ensure that the work products can be uniquely identified and reproduced in a controlled manner at any time and that the relations and differences between earlier and current versions can be traced. The CM process shall comply with the specific requirements for software development according to ISO/IEC 12207. There are no CM requirements relevant to software tools.

Section 8 of ISO 26262-8 contains requirements for change management to analyze and control changes to safety-related work products throughout the safety life cycle. Change management ensures the systematic planning, control, monitoring, implementation, and documentation of changes, while maintaining the consistency of each work product. Potential impacts on functional safety are assessed before changes are made and responsibilities for the decision-

making processes are assigned to the parties involved. There are no change management requirements relevant to software tools.

Section 9 of ISO 26262-8 contains requirements for verification to ensure that the work products comply with their requirements. Verification is applicable to the concept phase, the product development phase, and the production and operation phase. Verification planning shall be carried out for each phase and shall address the tools used for verification and the verification environment. The evaluation of the verification results shall contain the configuration of the verification environment and verification tools used, and the calibration data used during the evaluation.

Section 10 of ISO 26262-8 contains requirements for documentation with an objective to develop a documentation management strategy for the entire safety life cycle to facilitate an effective and repeatable documentation management process. The documentation can take various forms and structures (e.g., paper, electronic media, and databases) and tools can be used to generate documents automatically. However, Section 10 does not contain any requirements for the selection, verification, validation, or use of such tools.

Section 11 of ISO 26262-8 contains requirements for determining the level of confidence in the use of software tools. Section 11 of ISO 26262-8 provides criteria to determine the required level of confidence, if necessary, and provides means for the qualification of the software tool when applicable. When a software tool used in the development of a system or its software or hardware elements supports or enables a tailoring of activities and tasks of the safety life cycle required by ISO 26262 and the output of the software tool has not been examined or verified, then a software tool confidence level (TCL) shall be determined. The first step in determining the TCL is to analyze and evaluate the intended usage of the software tool to select a tool impact (TI) class. A class of TI1 applies when there is no possibility that a malfunction of the software tool can introduce or fail to detect errors in a safety-related item or element being developed. A class of TI2 applies in all other cases. The second step in determining the TCL is to analyze and evaluate the intended usage of the software tool to select a tool error detection (TD) class. A class of TD1 applies if there is a high degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected. A class of TD2 applies if there is a medium degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected. A class of TD3 applies in all other cases. Prevention or detection of erroneous output can be accomplished through process steps, redundancy in tasks, redundancy in software tools, or by rationality checks within the software tool itself. TD3 typically applies if there are no systematic measures in the development process available such that erroneous output can only be detected randomly. The third and last step in determining the TCL is to use the TI and TD classes and the following table to determine the TCL.

Table 2.4 – Determination of the Tool Confidence Level (TCL)

		Tool error detection		
		TD1	TD2	TD3
Tool impact	T11	TCL1	TCL1	TCL1
	T12	TCL1	TCL2	TCL3

The TCL is then used along with the ASIL to select an appropriate combination of four methods of qualifying the software tool. A software tool classified at TCL1 needs no qualification. A software tool classified at TCL2 or TCL3 must be qualified using an appropriate combination of four highly recommended (++) or recommended (+) methods depending on ASIL as shown in the following two tables. A discussion of the four available methods is provided in the next paragraph. ISO 26262-8 states that tables with a number followed by a letter in the leftmost column are alternative entries and an “appropriate” combination of methods shall be applied in accordance with the ASIL independent of whether the methods are listed in the table or not. An ISL observation, based on our analysis of the tables, is that the requirements for table use are vague because the term “appropriate” is not clearly defined and different organizations could select different combinations of methods and consider them appropriate. The requirements also do not satisfy verbatim compliance because of the possibility that different organizations can select different combinations of qualification methods and also because methods not recommended in the tables can be used. For clarity and to allow verbatim compliance, the standard should identify all acceptable methods of qualification in the tables and clearly indicate which combination of methods are appropriate.

Table 2.5 – Qualification of Software Tools Classified TCL3

Methods		ASIL			
		A	B	C	D
1a	Increased confidence from use	++	++	+	+
1b	Evaluation of the tool development process	++	++	+	+
1c	Validation of the software tool	+	+	++	++
1d	Development in accordance with a safety standard	+	+	++	++

Table 2.6 – Qualification of Software Tools Classified TCL2

Methods		ASIL			
		A	B	C	D
1a	Increased confidence from use	++	++	++	+
1b	Evaluation of the tool development process	++	++	++	+
1c	Validation of the software tool	+	+	+	++
1d	Development in accordance with a safety standard	+	+	+	++

Increased Confidence From Use

A software tool can be qualified by relying on prior use. A software tool shall only be argued as having increased confidence from use if evidence is provided that:

- (1) The software tool has been used previously for the same purpose with comparable use cases and with a comparable operating environment and with similar functional constraints,
- (2) The justification for increased confidence from use is based on sufficient and adequate data obtained through accumulated amount of usage,
- (3) The specification of the software tool is unchanged, and
- (4) The occurrence of malfunctions and corresponding erroneous outputs of the software tool acquired during previous developments are accumulated in a systematic way.

The increased confidence from use argument is only valid for the considered version of the software tool.

Evaluation of the Tool Development Process

An evaluation of the software tool development process can be used to qualify a software tool by showing that the development process applied to the development of the software tool complies with an appropriate national or international standard.

Validation of the Software Tool

A software tool can be qualified by completing a validation process. Validation measures (i.e., tests designed to evaluate functional and non-functional quality aspects of the software tool) shall demonstrate that the software tool complies with its specified requirements. The malfunctions and their corresponding erroneous outputs of the software tool occurring during validation shall be analyzed together with information on their possible consequences and with measures to avoid or

detect them. Also, the reaction of the software tool to anomalous operating conditions shall be examined including foreseeable misuse, incomplete input data, incomplete update of the software tool, and use of prohibited combinations of configuration settings.

Development in Accordance With a Safety Standard

A software tool can be considered qualified if the software tool is developed in accordance with a safety standard such as ISO 26262, IEC 61508, or RTCA DO-178. ISO 26262-8 states that no safety standard is fully applicable to the development of software tools; therefore, a relevant subset of requirements of a safety standard can be selected.

An ISL observation, based on our analysis of the software tool requirements, is that ISO 26262 provides a broad coverage and an in-depth process for confirming the safety of software tools. With regard to coverage, ISO 26262 tool requirements apply to tools used in any software development process of a safety-related item or element beyond just the generation of the executable code. The in-depth process of ISO 26262 includes the assignment of a tool impact class, assignment of a tool error defect class, and determining the tool confidence level. However, the tool impact classes in ISO 26262 do not distinguish between verification tools and development tools; therefore, the qualification requirements of verification and development tools are identical even though development tools can directly introduce errors into the final product and verification tools cannot. Even though ISO 26262 contains an in-depth process for determining the qualification requirements of software tools, it does not require qualification or specify requirements for software tools in the TCL1 class. As a minimum, the TCL1 class of software tools should be verified and validated to ensure the tool performs its intended function and performs the function correctly even if the software tool has no impact on the safety-related item or element. ISO 26262 also states that no safety standard is fully applicable to the development of software tools. However, RTCA DO-330 is a standard that is fully applicable to the development of software tools, therefore, this statement in ISO 26262 is no longer valid. Developing a software tool in accordance with the requirements of RTCA DO-330 would be acceptable for safety-related software tools.

Section 12 of ISO 26262-8 contains requirements for the qualification of software components with an objective of providing evidence for their suitability for re-use to avoid the need for re-development of software components with similar or identical functionality. Software components addressed by Section 12 of ISO 26262-8 include COTS software and in-house components already in use. The only requirement relevant to software tools is that the specification of the software component must include a description of the integration including development tools required to integrate and use the software component.

Section 13 of ISO 26262-8 contains requirements for the qualification of hardware components with an objective of providing evidence of the suitability of the functional behavior and operational limitations of intermediate level hardware components and parts for their use as part of items, systems, or elements of a safety-related system. The only requirement relevant to

tools is that the hardware qualification plan shall describe the necessary tools and equipment enabling the qualification strategy.

Section 14 of ISO 26262-8 contains guidance for the proven in use argument that is an alternate means of compliance with ISO 26262 regarding the reuse of existing items or elements when field data is available. The proven in use argument is not applicable to software tools.

2.6.2 Summary

ISO 26262 contains a probabilistic risk-based process for safety-related system and software development for the automotive industry. ISO 26262 contains requirements and guidance for software tool selection, use, documentation, and qualification. ISO 26262 also provides a comprehensive process for determining the confidence level in software tools used in the software development process. ISO 26262 provides comprehensive, well-organized software tool requirements and guidance that would form a sound technical basis for developing new regulatory guidance for the commercial nuclear power industry. However, the standard lacks a comprehensive review and approval process, the probabilistic risk-based process for designing road vehicle safety-related systems may not be appropriate for commercial nuclear power safety-related software development since software failures are impossible to quantify, and the software tool qualification requirements are the same for software development tools and software verification tools even though defects in software development tools can have a more significant impact on the software product than verification tools.

ISO 26262 is a ten-part international standard that is an automotive specific implementation of IEC 61508 that provides requirements for the functional safety of road vehicles. ISO 26262 provides an automotive safety life cycle and supports tailoring the necessary activities during those life cycle phases. ISO 26262 applies to all development activities during the safety life cycle of safety-related electrical or electronic systems installed in production passenger cars. ISO 26262 also provides an automotive-specific risk-based approach to determine ASILs and uses the ASIL to specify applicable requirements of ISO 26262 to avoid unreasonable residual risk. ISO 26262 requires software to be assigned to one of four ASIL categories from ASIL A (lowest) to ASIL D (highest) on the basis of the system ASIL and the level of risk associated with the use of the software in the system. A higher software ASIL corresponds to a higher risk resulting from software failure.

Two of the ten parts of ISO 26262 contains requirements and guidance related to software tools. ISO 26262-6 specifies requirements for software development and ISO 26262-8 specifies requirements for supporting processes including establishing confidence in the use of tools. ISO 26262-6 and ISO 26262-8 specify the requirements for all life cycle phases of safety-related software development and the requirements vary depending on the assigned software ASIL.

ISO 26262-6 contains a few high-level requirements for software tools used in the safety-related software development processes. Some of the software tool requirements are dependent on the ASIL associated with the software or system. ISO 26262-6 requires tools to be consistent across all the sub-phases of the tailored software life cycle, be compatible with the system and hardware development phases such that the required data can be transformed correctly, and

have guidelines for their application. These high-level requirements are not dependent on the ASIL. ISO 26262-6 also contains guidance as a function of ASIL for selecting a suitable modeling or programming language. The modeling and programming language should have unambiguous syntax and semantics, support for embedded real time software and runtime error handling, and support for modularity, abstraction, and structured constructs. ISO 26262-6 states that assembly languages can be used for those parts of the software where the use of high-level programming languages is not appropriate, such as low-level software with interfaces to hardware, interrupt handlers, or time-critical algorithms. An ISL observation, based on our analysis of ISO 26262-6, is that a requirement for software tool training should be added and the allowed use of assembly language should be reconsidered based on the additional complexity of code inspections, bi-directional tracing of code to requirements, and the specialized skill set needed to write and perform QA on assembly language code.

ISO 26262-8 contains a few high-level requirements and guidance for software tools used in a few of the automotive safety life cycle processes. ISO 26262-8 requires that the developer and supplier use compatible tools when the supplier shares some of the responsibility for the safety of a developed item. ISO 216262-8 also requires that the verification plan address software tools to be used during the verification process. ISO 26262-8 recommends but does not require the use of requirements specification software tools to specify and verify safety requirements.

ISO 26262-8 contains requirements for establishing an adequate level of confidence in the use of software tools dependent on the ASIL associated with the software or system. ISO 26262-8 requires that a tool confidence level (TCL) be determined when a software tool supports or enables the tailoring of activities and tasks of the safety life cycle and the output of the software tool has not been examined or verified. The software TCL is determined by first assigning the software tool to a tool impact (TI) class of TI1 or TI2 and then evaluating the intended usage of the software tool and selecting a tool error detection (TD) class of TD1, TD2, or TD3. Software tools are assigned a tool impact class of TI1 only when the tool cannot introduce or fail to detect errors in the safety-related item being developed. All other tools are assigned a tool impact class of TI2. The tool error detection classes of TD1, TD2, and TD3 are assigned based on the degree of confidence that a tool malfunction and erroneous tool output can be prevented or detected. Based on the tool impact and tool error detection, a tool confidence level of TCL1, TCL2, or TCL3 is assigned to the tool. The TCL is then used along with the ASIL to select an appropriate combination of four methods of qualifying the software tool. The four qualification methods are: confidence from prior tool use, evaluation of the tool development process, tool validation, and tool development in accordance with a safety standard. An ISL observation, based on our analysis of ISO 26262-8, is that the requirements for software tool qualification are not specific enough to ensure a consistent methodology within the automotive industry.

ISO 26262 contains an automotive-specific safety-related software development process based on the use of hazard analysis to determine risks and a risk assessment to determine mitigation strategies. Based on the analysis of ISO 26262, the following essential elements of the document relative to software tools have been determined.

- (1) An automotive-specific risk-based approach is implemented to determine automotive safety integrity levels (ASIL A through D).
- (2) The ASIL is used to specify applicable requirements of ISO 26262 so as to avoid unreasonable residual risk.
- (3) Tools used in the software development process shall be consistent across all the sub-phases of the tailored software life cycle and be compatible with the system and hardware development phases.
- (4) For each sub-phase of software development, corresponding tools shall be selected including guidelines for their application.
- (5) Assembly languages can be used for those parts of the software where the use of high-level programming languages is not appropriate, such as low-level software with interfaces to hardware, interrupt handlers, or time-critical algorithms.
- (6) The vehicle manufacturer and the supplier shall specify a developer interface agreement that discusses the supporting processes and tools, including interfaces, to assure compatibility between vehicle manufacturer and supplier.
- (7) When a software tool used in the development of a system or its software or hardware elements supports or enables a tailoring of activities and tasks of the safety life cycle and the output of the software tool has not been examined or verified, then a software tool TCL shall be determined.
- (8) An analysis and evaluation of the intended usage of the software tool is performed to select a TI class based on the possibility that a malfunction of the software tool can introduce or fail to detect errors in a safety-related item or element being developed.
- (9) An analysis and evaluation of the intended usage of the software tool is performed to select a TD class based on the degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected.
- (10) The TCL is determined from the TI and TD classes.
- (11) The TCL is used along with the ASIL to select an appropriate combination of four methods of qualifying the software tool including: confidence from use, evaluation of the tool development process, validation, and development in accordance with a safety standard.

Based on the analysis of ISO 26262, ISL has made the following important observations and recommendations.

- (1) ISO 26262 should require an appropriate amount of training in the use of a tool so that the guidelines for tool use can be safely implemented.
- (2) Use of assembly language for programming low-level software would introduce several unique problems related to the verification, validation, and testing of the software.
- (3) Bi-directional tracing of assembly language code to requirements specifications is difficult.
- (4) Assembly language code inspection is more difficult than code inspection for high-level programming languages and requires a more specialized set of skills which may inhibit QA activities and increase the probability of defects in the code.
- (5) The qualification requirements of verification and development tools are identical because the TI classes in ISO 26262 do not distinguish between verification tools and development tools.
- (6) ISO 26262 does not make use of the required effectiveness of each tool qualification method as a function of ASIL as implemented in IEC 61508.
- (7) ISO 26262 should identify all acceptable methods of qualification and clearly indicate which combination of methods are appropriate.
- (8) ISO 26262 contains an in-depth process for determining the qualification requirements of software tools, but it does not require qualification or specify requirements for software tools in the TCL1 class. As a minimum, the TCL1 class of software tools should be verified and validated to ensure the tool performs its intended function and performs the function correctly even if the software tool has no impact on the safety-related item or element.
- (9) RTCA DO-330 is a standard that is fully applicable to the development of software tools but is not recognized by ISO 26262.
- (10) ISO 26262 lacks a comprehensive review and approval process.
- (11) The probabilistic risk-based process for designing road vehicle safety-related systems may not be appropriate for commercial nuclear power safety-related software development since software failures are impossible to quantify.

2.7 Railway Industry Analysis

Prevailing railway control systems are similar to air traffic control systems by relying on dispatchers at central locations to track the location of trains and signal train operators when it is

safe to proceed onto a stretch of track. High-speed rail networks use electronic train control systems including communications systems, global positioning systems, on-board computers with digitized maps, and central control system computers to monitor and control train movements. This computer-based technology improves efficiency and safety through better communication and reducing the probability of human error in the operation of trains. Several European standards address the reliability, availability, maintainability, and safety of railway applications. These standards define processes based on a system life cycle for managing reliability, availability, maintainability, and safety for individual and combined sub-systems and components within major systems including those containing software.

2.7.1 Document Analysis

This section discusses an analysis of BS EN 50128:2011 that defines software requirements for control and protection systems for the railway industry. This is a British standard that is a railway specific implementation of IEC 61508. The analysis discussion focuses on the use, selection, requirements, verification, validation, and qualification of software tools used in the railway industry.

2.7.1.1 BS EN 50128

BS EN 50128 is a United Kingdom implementation of European Standard EN 50128 and a railway specific implementation of International Standard IEC 61508. BS EN 50128 is part of a group of related standards. The others are BS EN 50126-1 and BS EN 50129. BS EN 50126-1 addresses high-level system issues and BS EN 50129 addresses the approval process for systems. These related standards require that a systematic approach be taken to identifying hazards, assessing risks, and identifying the necessary risk reduction measures to mitigate the risk. BS EN 50128 concentrates on the methods to provide software that meets the demands for safety integrity assigned by the related standards.

BS EN 50128 specifies the process and technical requirements for the development, deployment, and maintenance of any safety-related software for programmable electronic systems intended for railway control and protection applications. BS EN 50128 presents requirements for each major software life cycle process including software management and organization, software assurance, software development, and software deployment and maintenance. Requirements in the software management and organization process cover personnel roles, responsibilities, and competence. Requirements in the software assurance process cover testing, verification, validation, assessment, QA, modification and change control, and support tools and languages. Requirements in the software development process cover requirements specification, architecture and design, component design, component implementation and testing, integration, and final validation. The requirements for support tools, which are the focus of this survey and analysis, are presented in Section 6.7 of BS EN 50128 and cover new tools and pre-existing tools.

BS EN 50128 acknowledges that the system and software development state-of-the-art is such that neither the application of fault avoidance and fault detection measures nor the application of fault tolerance approaches can guarantee the absolute safety of the software. The standard

also acknowledges that there is no way to prove the absence of faults in reasonably complex safety-related software, especially the absence of specification and design faults. The risk of faults can be minimized by applying good software development principles during the development of high-integrity software. Sections 1 through 3 of BS EN 50128 contain introductory material and do not discuss software tools.

The key concept of BS EN 50128, Section 4, is that of software SIL. The requirements in this section state that the software safety integrity shall be specified as one of five levels, from SIL0 (the lowest) to SIL4 (the highest). A higher software SIL is associated with a higher risk resulting from software failure. The required software SIL shall be decided and assessed at a system level, on the basis of the system SIL and the level of risk associated with the use of the software in the system. At least the SIL0 requirements shall be fulfilled for the software part of functions that have a safety impact below SIL1. This is because uncertainty is present in the evaluation of the risk, and even in the identification of hazards; therefore, in the face of uncertainty, it is prudent to aim for a low level of safety integrity, represented by SIL0, rather than none. Consistent with IEC 61508, BS EN 50128 identifies techniques and measures for meeting the requirements of each software life cycle phase. For each life cycle phase, tables in Annex A of BS EN 50128 specify those techniques and measures necessary to achieve the requirements of the standard and identify a technique or measure as mandatory, highly recommended, recommended, not recommended, or the table makes no recommendation for or against the technique or measure. These recommendations vary by software SIL; however, BS EN 50128 does not utilize all five levels of safety integrity. Instead, the requirements for SIL1 and SIL2 are the same for each technique or measure. Similarly, each technique or measure has the same requirements at SIL3 and SIL4. Therefore, effectively, there are recommendations for techniques and measures for only three SILs in BS EN 50128. An ISL observation, based on an analysis of software SIL, is that the software SIL is not used to determine the requirements for tool qualification, selection, or use, and that SIL is only used to specify techniques and measures to satisfy the software development requirements of this standard.

Section 5 of BS EN 50128 discusses software management and organization with an objective of ensuring that all personnel who have responsibilities for the software are organized, empowered, and capable of fulfilling their responsibilities. The assignment of roles to persons involved in the software development project varies by software SIL. Higher software SILs require more personnel and more independence between personnel. Section 5 requires that a life cycle model be selected and documented in the QA plan. Two life cycle models are provided as examples; however, the use of either is not mandatory. Any suitable life cycle model may be utilized provided it meets all the requirements of BS EN 50128. Section 5 contains no requirements relevant to tools.

Section 6 of BS EN 50128 discusses requirements for software assurance. Section 6.1 of BS EN 50128 contains requirements for software testing. The objective of software testing is to ascertain the performance of software to the extent achievable by the selected test coverage. The only software testing requirements relevant to tools are that any tools used for testing shall

be shown to be suitable for the purpose and the test specification shall document the test environment, tools, configuration, and programs used for testing.

Section 6.2 of BS EN 50128 contains requirements for software verification. The objective of software verification is to examine and ascertain that the output of all life cycle phases fulfill the requirements and plans of that phase with respect to completeness, correctness, and consistency. The only software verification requirement relevant to tools is that the software verification plan must document all the criteria, techniques, and tools to be used in the verification process.

Section 6.3 of BS EN 50128 contains requirements for software validation. The objective of software validation is to demonstrate by analysis or testing that the software meets the defined software SIL, fulfils the software requirements, and is fit for its intended application. There are no software validation requirements relevant to tools.

Section 6.4 of BS EN 50128 contains requirements for software assessment. The objective of software assessment is to demonstrate that the software is of the defined software SIL and is fit for its intended application. There are no software assessment requirements relevant to tools.

Section 6.5 of BS EN 50128 contains requirements for software QA. The objective of software QA is to identify, monitor, and control all technical and management activities to provide a qualitative defense against systematic faults. The only software QA requirements relevant to tools are that the software QA plan shall specify or reference the methods, measures, and tools for QA according to the allocated SILs and the CM system shall cover the software development environment including all the tools, translators, data, and test files necessary for the reproducibility of the development.

Section 6.6 of BS EN 50128 contains requirements for modification and change control. The purpose of modification and change control is to ensure that the software performs as required while preserving the software safety integrity and dependability when modifying the software. There are no modification and change control requirements relevant to tools.

Section 6.7 of BS EN 50128 discusses requirements for support tools and languages. The objective of the requirements in this section of the standard is to provide evidence that potential failures of tools do not adversely affect the integrated toolset output in a safety-related manner that is undetected by technical or organizational measures outside the tool. Tools are categorized into three classes: T1, T2, and T3. Class T1 tools do not generate output which can directly or indirectly contribute to the executable code or data required by the software (e.g., text editors, requirement or design support tool with no automatic code generation capability, configuration control tools). Class T2 tools support the test or verification of the design or executable code and errors in the tool can fail to reveal defects but cannot directly create errors in the executable software (e.g., test harness generator, test coverage measurement tool, static analysis tool). Class T3 tools generate output that can directly or indirectly contribute to the executable code or data of the safety-related system (e.g., compiler, linker, tool to change set-points during system operation).

The requirements for tools depend on the class of the tool. Class T1 tools have fewer and less stringent requirements. Class T2 tools have more requirements than class T1 tools but less than class T3 tools. Class T3 tools have the most number of requirements and the most stringent requirements. A common requirement of all tools is that they should be selected as a coherent part of the software development activities and the outputs of one tool should have suitable content for the subsequent tool. No other requirements in BS EN 50128 apply to class T1 tools.

Additional requirements for class T2 and T3 tools are that the tools shall be justified including identification of potential failures and measures to avoid or handle such failures, the tools need a specification or manual which clearly defines the behavior of the tool, the tools must be maintained under CM to ensure that only justified versions are used, and new tool versions need to be justified. Justification for new tools may rely on evidence provided for the earlier version of the tool provided the functional differences will not affect the entire toolset and the new version is unlikely to contain significant new, unknown faults based on credible identification of the changes made and on an analysis of the V&V actions performed on the previous tool version.

Class T3 tools require validation or an alternate form of evidence that the output of the tool conforms to the specification of the output or failures in the output are detected. Alternate forms of evidence include performing the process of the tool manually, using a combination of history of successful use in similar environments and for similar applications, using diverse redundant code to detect and control failures, or showing compliance with the SIL derived from the risk analysis. If a class T3 tool cannot be validated or evidence is not available to confirm the behavior of the tool, then effective measures to control failures of the safety-related software that result from tool faults must be implemented (e.g., diverse redundant code).

Two additional class T3 tool requirements relate to translators, which convert a software design representation from one abstraction level to another level. The first class T3 tool requirement related to translators is that translators must be evaluated for fitness of purpose including evaluation against international or national standards, must match the characteristics of the application, must contain features that facilitate detection of design or programming errors, and must support features that match the design method. The second class T3 tool requirement related to translators is that the suitability of automatic translation must be evaluated at the point in the development life cycle where development support tools are selected. The evaluation of a class T3 tool may be performed for a specific application project or for a class of applications. If evaluated for a class of applications, all the necessary information on the tool shall be available to the user of the tool and the evaluation of the tool for the specific project is reduced to a simple check for general suitability and compliance to the tool specification or manual. A validation suite may also be used to evaluate the fitness for purpose of a translator.

An ISL observation, based on our analysis of tool requirements in Section 6.7 of BS EN 50128, is that more requirements should be specified for class T1 tools because there are effectively no requirements other than class T1 tools should be selected to be a coherent part of the development environment. The standard does not require verification, validation, or evidence of

suitability of class T1 tools. Class T1 tools that generate a requirements specification, design specification, or are used for CM should, as a minimum, be verified and validated to ensure the tool performs its intended function and performs the function correctly especially since this standard acknowledges that specification and design faults are potential contributors to the uncertainty in the safety of software. Another option is to define a new class of tools with more requirements for verification, validation, and evidence of suitability appropriate for tools that are used to generate a requirements specification, design specification, or used for CM.

Another ISL observation, based on our analysis, is that the requirements for new tools should be strengthened beyond the credible identification of changes and analysis of the V&V activities performed on the original tool version. Changes incorporated into new tool versions are difficult to assess, especially the impact of the changes on the output. New class T2 and T3 tool versions should be subjected to some amount of re-verification and revalidation to better assess the changes. ISL further observes that evaluating a class T3 tool for an entire class of applications and only checking its general suitability for the specific project may not be sufficient in a safety-related application. Each class T3 tool should be evaluated for the specific project application to ensure the safety of the tool.

Section 7 of BS EN 50128 contains requirements for generic software development including requirements for generic software documentation, choosing a life cycle, software requirements, architecture and design, component design, component implementation and testing, software and hardware integration, and overall software testing and final validation. With respect to tools, the only requirements in Section 7 of BS EN 50128 are high-level requirements that the techniques, measures, and tools chosen form a set which satisfies the software requirements specification at the required software and system SIL and that the test environment including tools, support software, and configuration description be documented in the software and hardware integration test specification.

The structure of BS EN 50128 supports the development of railway specific safety-related software using two sets of requirements. Generic software is developed using the requirements in Section 7 of BS EN 50128 and then the railway specific safety-related software is customized using application specific data and algorithms based on the requirements in Section 8 of BS EN 50128. The use of application data and application algorithms allows approved generic software to be customized with the individual requirements for each specific application.

Section 8 of BS EN 50128 contains requirements for development of application data or algorithms specific to the railway application including requirements for the application development process, application requirements specification, architecture and design, application data and algorithm production, application integration and testing acceptance, application validation and assessment, application preparation procedures and tools, and development of generic software. Section 8 of BS EN 50128 contains several high-level requirements that pertain to tools. The first requirement is that the procedures and tools used for application development shall be appropriate to the system SIL as determined by the function for which they are developed. A second requirement is that the procedures and application tools to be used in the application development process shall be specified in the

application preparation plan. A third requirement is that the application preparation plan shall include V&V activities to ensure that the application tools and the generic software are compatible with each other and with the specific application, and to provide evidence that their application conditions are met. A fourth requirement is that the application preparation plan shall also include V&V activities to ensure that the application data and algorithms are complete, correct, and compatible with each other and with the generic application, and to provide evidence that the application conditions of the generic application are met. The requirement further states that the V&V activities and evidence can be replaced by V&V performed on the tools that produce the application data and algorithms.

Additional requirements for tools state that a risk analysis shall be carried out covering the application development process, including the application tools and procedures, and that the application preparation plan shall define a tool class for any hardware or software tools used in the application preparation life cycle. The final requirements relevant to tools deal directly with application preparation tools. One requirement states that for each new type of system configured by application data and algorithms, tools shall be developed in accordance with BS EN 50128 in parallel with the generic software and hardware for the system. Further, any compilation process shall be validated and assessed since specialized compilers are usually necessary for the data and algorithm conversion. An additional requirement is that the application verification report shall demonstrate the coverage and enforcement of the application conditions of the generic software and application tools. Finally, a document shall be published prior to the final validation phase of the application tools that references the user manuals of the application tools and addresses any constraints on the application data or algorithms such as imposed architecture or coding rules to meet the SILs.

An ISL observation, based on our analysis of tool requirements in the application development phase, is that some ambiguity exists in the requirements. Section 8 of BS EN 50128 requires that application development tools be developed in accordance with BS EN 50128, be assigned a tool class; and be verified and validated. However, some application development tools are assigned as tool class T1 and class T1 tools are not required to be verified or validated based on the requirements of Section 6.7 of BS EN 50128. The tool requirements in Section 6.7 and Section 8 of BS EN 50128 should either be consolidated or rewritten for overall consistency and coverage. One alternative is that all tools used in the application specific development process should be classified as T3 so that the tools are treated to the proper level of safety based on the output produced by these tools.

Section 9 of BS EN 50128 contains requirements for software deployment and maintenance with an objective to ensure that the software performs as required when it is deployed in the final environment of application and when making corrections, enhancements, or adaptations to the software itself. The only requirement relevant to tools is that maintenance shall be performed with at least the same level of expertise, tools, documentation, planning, and management as for the initial development of the software.

2.7.2 Summary

BS EN 50128 contains a probabilistic risk-based process for safety-related system and software development for the railway industry. BS EN 50128 contains requirements and guidance for software tool selection, use, documentation, and V&V. BS EN 50128 provides comprehensive software tool requirements and guidance including the assignment of software to SILs and the assignment of tools to classes that would form a reasonable technical basis for developing new regulatory guidance for the commercial nuclear power industry. However, the standard lacks a comprehensive review and approval process, lacks qualification methods and techniques based on tool class, and the probabilistic risk-based process for designing railway safety-related systems may not be appropriate for commercial nuclear power safety-related software development since software failures are impossible to quantify.

BS EN 50128 is a railway-specific implementation of IEC 61508 and specifies technical requirements for the development of safety-related software for railway control and protection systems. This standard requires a systematic approach for safety-related software development to identify hazards, assess risks, and identify risk reduction measures. BS EN 50128 requires software to be assigned to one of five SILs from SIL0 (lowest) to SIL5 (highest) on the basis of the system SIL and the level of risk associated with the use of the software in the system. A higher software SIL corresponds to a higher risk resulting from software failure. Tables in Annex A of BS EN 50128 specify the techniques and measures for meeting the requirements for all life cycle phases of safety-related software development. The techniques and measures recommended vary depending on the assigned software SIL. Only three SILs are actually used and the safety-related software SIL and the tables of recommendations for measures and techniques to satisfy the requirements of BS EN 50128 do not impact the selection, use, or qualification of software tools used in the safety-related software development processes.

BS EN 50128 contains several requirements for software tools used in the safety-related software development life cycle process but the requirements are independent of the safety-related software SIL developed using the tool and only depend on the tool functionality. Tools are assigned to one of three classes and the tool requirements vary by class. Class T1 tools do not generate output that impacts the safety-related executable code (i.e., text editors, requirements specification tools) and have few requirements. Class T2 tools support safety-related software testing and verification (i.e., test coverage tools, static analysis tools). Class T2 tools have more requirements to justify the suitability of the tools and for maintaining CM. Class T2 tool errors may fail to detect defects in the safety-related software but cannot introduce errors. Class T3 tools contribute to the safety-related executable code or safety-related system data (i.e., automatic code generators, compilers, linkers). Class T3 tools have the most requirements including validation or alternate evidence that the output of the tool conforms to the tool specification. If a class T3 tool cannot be validated or its output cannot be confirmed by alternate means, then the safety-related software must implement measures to control failures of the software resulting from faults in the tool. An ISL observation, based on our analysis of BS EN 50128 tool requirements, is that some of the class T1 tools (i.e., design support tools and requirements specification tools) should be subject to more requirements including validation

and verification since errors in these tools can ultimately impact the safety-related software design and functionality. Another ISL observation is that the requirements for class T2 tools should be expanded to include V&V and the requirements for class T3 tools should be expanded to require project-specific application evaluation of tools rather than relying on a high-level tool evaluation for an entire class of applications and a check of general suitability for the specific application.

BS EN 50128 contains a railway-specific safety-related software development process based on the use of hazard analysis to determine risks and a risk assessment to determine mitigation strategies. Based on the analysis of BS EN 50128, the following essential elements of the document relative to software tools have been determined.

- (1) The software safety integrity shall be specified as one of five levels, from SIL0 (the lowest) to SIL4 (the highest).
- (2) The required software SIL shall be decided and assessed at a system level, on the basis of the system SIL and the level of risk associated with the use of the software in the system.
- (3) Any tools used for testing shall be shown to be suitable for the purpose and the test specification shall document the test environment, tools, configuration, and programs used for testing.
- (4) The software verification plan must document all the criteria, techniques, and tools to be used in the verification process.
- (5) The software QA plan shall specify or reference the methods, measures, and tools for QA according to the allocated SILs.
- (6) The CM system shall cover the software development environment including all the tools, translators, data, and test files necessary for the reproducibility of the development.
- (7) The objective of requirements in BS EN 50128 is to provide evidence that potential failures of tools do not adversely affect the integrated toolset output in a safety-related manner that is undetected by technical and organizational measures outside the tool.
- (8) Tools are categorized into three classes: T1, T2, and T3. The requirements for tools differ depending on the class of the tool.
- (9) All tools should be selected as a coherent part of the software development activities and the outputs of one tool should have suitable content for the subsequent tool.
- (10) Class T2 and T3 tools shall be justified including identification of potential failures and measures to avoid or handle such failures, shall have a

specification or manual which clearly defines the behavior of the tool, must be maintained under CM to ensure that only justified versions are used, and new tool versions need to be justified.

- (11) Class T3 tools require validation or an alternate form of evidence that the output of the tool conforms to the specification of the output or failures in the output are detected.
- (12) If a class T3 tool cannot be validated or evidence is not available to confirm the behavior of the tool, then effective measures to control failures of the safety-related software that result from tool faults must be implemented (e.g., diverse redundant code).
- (13) Translators must be evaluated for fitness of purpose including evaluation against international or national standards, must match the characteristics of the application, must contain features that facilitate detection of design or programming errors, and must support features that match the design method.
- (14) Suitability of automatic translation must be evaluated at the point in the development life cycle where development support tools are selected.
- (15) The evaluation of a class T3 tool may be performed for a specific application project or for a class of applications. If evaluated for a class of applications, all the necessary information on the tool shall be available to the user of the tool and the evaluation of the tool for the specific project is reduced to a simple check for general suitability and compliance to the tool specification or manual.
- (16) The techniques, measures, and tools chosen must form a set which satisfies the software requirements specification at the required software and system SIL.
- (17) The test environment including tools, support software, and configuration description must be documented in the software and hardware integration test specification.
- (18) The application preparation plan shall include V&V activities to ensure that the application tools and the generic software are compatible with each other and with the specific application.

Based on the analysis of BS EN 50128, ISL has made the following important observations and recommendations.

- (1) The software SIL is not used to determine the requirements for tool qualification, selection, or use, and the SIL is only used to specify techniques and measures to satisfy the software development requirements.
- (2) BS EN 50128 effectively only uses three of the five levels of safety integrity.

- (3) Class T1 tools that generate a requirements specification, design specification, or are used for CM should, as a minimum, be verified and validated to ensure the tool performs its intended function and performs the function correctly especially since BS EN 50128 acknowledges that specification and design faults are potential contributors to the uncertainty in the safety of software.
- (4) The requirements for new Class T2 and T3 tools should be strengthened beyond the credible identification of changes and analysis of the V&V activities performed on the original tool version.
- (5) Evaluating a class T3 tool for an entire class of applications and only checking its general suitability for the specific project may not be sufficient in a safety-related application. Each class T3 tool should be evaluated for the specific project application to ensure the safety of the tool.
- (6) Tool requirements in the application development phase are ambiguous because BS EN 50128 requires that application development tools be developed in accordance with BS EN 50128, be assigned a tool class; and be verified and validated; however, some application development tools are assigned as tool class T1 and class T1 tools are not required to be verified or validated.
- (7) BS EN 50128 lacks a comprehensive review and approval process.
- (8) BS EN 50128 lacks qualification methods and techniques based on tool class.
- (9) The probabilistic risk-based process for designing railway safety-related systems may not be appropriate for commercial nuclear power safety-related software development since software failures are impossible to quantify.

2.8 International Atomic Energy Agency Analysis

The International Atomic Energy Agency (IAEA) serves as the world's forum for scientific and technical cooperation in the nuclear field. The IAEA promotes safe, secure, and peaceful nuclear technologies. Three main areas of work include safety and security, science and technology, and safeguards and verification. The IAEA is a leading publisher in nuclear science and technology, with titles on nuclear and radiological safety, emergency response, nuclear power, nuclear medicine, nuclear waste management, nuclear law, and safeguards. Key publications include the IAEA safety standards, which detail the principles of safety for protection against ionizing radiation, and IAEA safety reports, which describe good practices and give practical examples and detailed methods that can be used to meet safety requirements. Within the IAEA safety standards series of publications are safety fundamentals, safety requirements, and safety guides. IAEA safety fundamentals present basic objectives, concepts, and principles of safety. IAEA safety requirements establish requirements that are mandatory to ensure safety governed by the safety fundamentals. IAEA safety guides recommend actions, conditions, or procedures for meeting safety requirements that are not

mandatory but should be implemented as recommended or through an equivalent alternative measure.

2.8.1 Document Analysis

This section discusses an analysis of safety guides IAEA NS-G-1.1 and IAEA DS-431. IAEA NS-G-1.1 is the active safety guide that provides guidance for demonstrating the safety of software used in computer-based systems of commercial nuclear power plants. IAEA DS-431 is a draft safety guide intended to supersede IAEA NS-G-1.1. The analysis discussion focuses on the use, selection, verification, validation, and qualification of software tools used in the development of computer-based systems of commercial nuclear power plants.

2.8.1.1 IAEA NS-G-1.1

IAEA NS-G-1.1 is a safety guide that provides guidance for demonstrating the safety of software used in nuclear power plant computer-based systems. The safety guide includes guidance for software tools used in the computer-based system and software development life cycle. This safety guide is organized with several introductory sections followed by sections specifically dedicated to individual phases of the development life cycle.

Section 1 of IAEA NS-G-1.1 contains background information and defines the objectives, scope, and structure of the safety guide. The background information discusses the importance of safety in nuclear power plants that contain computer-based systems and states that the dependability of computer-based systems can only be predicted and demonstrated if a systematic, fully documented, and reviewable process is followed. The objective of the safety guide is to provide guidance on collecting evidence and preparing documentation for demonstrating the safety of software used in nuclear power plant computer-based systems. The scope of the guidance relates primarily to the software used in computer based systems and covers all phases of the system life cycle. The guidance is applicable to systems important to safety which covers safety systems and safety-related systems. The following definitions apply within this standard.

System important to safety is a system that is part of a safety group whose malfunction or failure could lead to radiation exposure of the site personnel or members of the public.

Safety system is a system important to safety, provided to ensure the safe shutdown of the reactor or the residual heat removal from the core, or to limit the consequences of anticipated operational occurrences and design basis accidents.

Safety-related system is a system important to safety that is not part of a safety system.

Note that these IAEA-NS-G-1.1 definitions are similar to those in NUREG/CR-6421, which distinguish between safety-related systems and safety systems, but they are inconsistent with the definitions in NRC regulations in which the terms safety and safety-related mean the same thing, and in which safety systems (i.e., safety-related

systems), are a subset of important-to-safety systems, which also include certain nonsafety (i.e., nonsafety-related) systems, the failure of which could impact one or more safety functions.

The main focus of safety guide IAEA NS-G-1.1 is on the preparation of documentation used to adequately demonstrate the safety and reliability of computer based systems important to safety. This safety guide applies to all types of software: pre-existing software or firmware (such as an operating system), software to be specifically developed for the project, or software to be developed from an existing pre-developed equipment family of hardware or software modules.

Section 2 of IAEA NS-G-1.1 provides recommendations on the technical considerations for computer based systems, addressing advantages and disadvantages of such systems, safety and reliability issues, and some organizational preconditions of the system development project. Advantages of computer-based systems are that they can facilitate the achievement of complex functions including improved monitoring of plant variables, improved operator interfaces, improved testing, and fault diagnosis with greater accuracy and stability. The disadvantages are that software implementation is more complex, more prone to design errors, more sensitive to small errors, and more difficult to test. Guidance for the development process is provided by recommending an organized and ordered step-by-step process with distinct phases. Products of each phase are verified against the requirements of the previous phase. Section 2 of IAEA NS-G-1.1 does not discuss software tools and provides no guidance for the use, verification, validation, review, or approval of software tools.

One discussion of interest in Section 2 of IAEA NS-G-1.1 relates to common mode failures. With respect to common mode failures, the safety guide states that since software faults are systematic rather than random in nature, common mode failure of computer-based safety systems employing redundant subsystems using identical copies of the software is a critical issue. Countermeasures include independence, diversity, and a comprehensive qualification strategy to protect against common mode failures. Even with these countermeasures, the degree of success and the benefits of these strategies is difficult to estimate when software is involved.

Section 3 of IAEA NS-G-1.1 provides an overview of the most relevant requirements for management of safety, guidance for determining design and development activities, guidance for management and QA activities, and guidance for adequate documentation. Diversity is one of the relevant requirements for the management of computer-based system safety. Software tool guidance relative to diversity is that the diversity of methods, languages, tools, and personnel should be taken into consideration along with the use of diverse functions and system components at different levels of the design. Additional guidance for software tools states that particular attention should be given to the use of automated tools in determining the necessary design and development activities. The guidance states that all tools used should be mutually compatible and qualified to a level commensurate with their function in the development of the software and in the safety demonstration. The techniques used to gain confidence in the tools should be specified and documented. Guidance for acceptable practices states that well

established methods, languages, and tools for software development should be used and methods, languages, and tools that are still at the research stage should not be used.

Section 4 of IAEA NS-G-1.1 provides recommendations on the planning phase of the system development project and describes the structure and content of the documentation associated with the development plan, QA program description, V&V plan, CM plan, and installation and commissioning plan. Software tool guidance is provided for all project plans except for the installation and commissioning plan.

Development Plan

The computer-based system development plan should identify and define the development process to be used on a particular project. Software tool guidance associated with the development plan states that tools to be used in the software development process should be identified in the development plan; selected to facilitate the proper application of the methods, standards, and procedures selected; and planned in advance for the entire project to help in the selection of an integrated set of tools. Further guidance states that tools should be properly qualified for their function in the development, management, or verification of the system and that the correctness of their output should be ensured by a tool certification process, by means of cross-verification, or by reverse engineering. The safety guide encourages tool use since they relieve staff from manual error prone tasks such as programming and verification. Tools also help to achieve a reproducible level of quality, in part guaranteed and demonstrated by the evidence provided by their past usage.

IAEA NS-G-1.1 states that one of the managerial tasks associated with the computer-based system development plan is assessing the adequacy of the facilities and tools available. With respect to personnel, the safety guide states that a plan should be prepared to ensure that those personnel involved in the development activities are competent in the use of design, programming, and analysis tools and methods. Records of their competence should be maintained and new members should be closely supervised until they have demonstrated their competence.

Quality Assurance Program

A QA program description should be prepared and implemented and should be available for regulatory review and approval before the project begins. A software QA plan should be produced at the outset of the project. Software tool guidance associated with QA is that a QA plan should be prepared that covers external suppliers and should identify the tools to be used on the project and the procedures for qualifying the tools.

Verification and Validation Plan

In the context of this safety guide, verification is the process of checking a product against the output of the preceding phase and validation is the process of checking a product against higher level requirements and goals. The team or teams performing

the V&V should be identified in the V&V plan and should be independent of the development team. Software tool guidance associated with V&V states that as part of technical independence, the work should be done using different techniques and tools.

Configuration Management Plan

This safety guide considers CM to be an extension of the development plan and QA program description. Software tool guidance associated with CM states that all items of software development, such as compilers, development tools, configuration files and operating systems, should be under CM control.

Section 5 of IAEA NS-G-1.1 contains computer system requirements for computer-based system development. Computer system requirements define the necessary behavior of the computer system and are based on the high-level plant design. Safety requirements based on accident and transient analyses are an essential element of the design. IAEA NS-G-1.1 encourages the use of tools to support the formalism necessary for the specifications and to assist the validation of the completeness and consistency of the computer system requirements. The only software tool relevant guidance states that tools used to assist the validation of the completeness and consistency of the computer system requirements should be validated to a level of confidence equivalent to the level required from the design process.

Section 6 of IAEA NS-G-1.1 contains guidance for designing the computer system by mapping system requirements into requirements to be satisfied by a combination of pre-existing or pre-developed software or firmware, hardware or application-specific integrated circuits, and software to be developed or produced by configuring pre-developed software. Section 6 of IAEA NS-G-1.1 does not discuss software tools and provides no guidance for the use, verification, validation, review, or approval of software tools.

Section 7 of IAEA NS-G-1.1 contains guidance for preparing software requirements. Software requirements are the subset of the computer system requirements that will ultimately be implemented as computer programs. Preparing the software requirements is a process that is closely tied to designing the computer system. The software requirements describe what the software component must do to meet the overall system requirements. Software tool guidance associated with preparing software requirements states that personnel responsible for preparing software requirements should be aware of the capabilities of the computers, tools, and similar existing systems so that it is feasible to fulfill the software requirements in the software design and the software implementation. Section 7 of IAEA NS-G-1.1 also addresses software requirements when automatic code generation tools or computer system requirement tools are used. The guidance states that if parts of the computer system design or code are generated by tools, then a separate software requirement document may be unnecessary for those parts if the computer system requirements are sufficiently detailed and formally documented. However, those parts of such computer system requirements from which code is generated or reused should be regarded as a statement of software requirements against which subsequent code should be verified. In addition, any separately compiled modules that are included by the code generator should be supported by separate documents for the software requirements. An ISL

observation, based on our analysis of the software requirement phase guidance, is that Section 7 of IAEA NS-G-1.1 does not address tools that automate the software requirement process by translating high-level system requirements into high-level software requirements. However, these types of tools are addressed in Section 9 of IAEA NS-G-1.1 as part of implementation phase guidance.

Section 8 of IAEA NS-G-1.1 contains guidance for the software design phase. Software design is the division of the software into a set of interacting modules. The software design should be well structured and understandable to the software implementation team, maintenance personnel, testing team, and regulators. The only software tool guidance in the software design phase is associated with implementation constraints. The guidance states that implementation constraints, such as the attributes required of the programming languages, compilers, subroutine libraries, and other supporting tools, should be documented and justified or be traceable to higher level requirements or constraints. An ISL observation, based on our analysis of the software design phase guidance, is that Section 8 of IAEA NS-G-1.1 does not address tools that automate the software design process by translating high-level software requirements into low-level design requirements. However, these types of tools are addressed in Section 9 of IAEA NS-G-1.1 as part of implementation phase guidance.

Section 9 of IAEA NS-G-1.1 contains guidance for the software implementation phase. The inputs to this phase are the system specifications and a set of interacting modules defined in the software design phase. The product of the software implementation phase is the source code, executable code, and the results of unit tests and module interface tests. Software tool guidance associated with the software implementation phase states that code can be produced from the system specifications using combinations of two distinct approaches. One approach is the classical development process through the stages of software requirements specification and software design. The second approach is the use of code generating tools which receive as input an application oriented description of the system with a high-level language. The choice between these two approaches depends on the tools and resources available to the project and should take into account trade-offs between design and the demonstration of the dependability of tools.

Additional software tool guidance in Section 9 of IAEA NS-G-1.1 states that only validated tools should be used, a suitable set of tools should be selected, translators should be thoroughly tested and available throughout the system lifetime, and tools used in the implementation phase should be compatible with tools used in other phases of software development. Guidance further states that if the translator used is not thoroughly tested, additional analysis and verification, manual or using another tool, should demonstrate that the translator is correct.

Final guidance for software tools in the software implementation phase deals with diversity and documentation. Software diversity can be achieved by using different working environments, tools, and run time support systems (e.g., operating systems and compilers). The choice of all tools used in the software implementation phase should be justified and properly documented along with the process by which the tools have been validated.

Section 10 of IAEA NS-G-1.1 contains guidance for verification and analysis. Verification is the process of ensuring that the products of each development phase (including any pre-existing software) satisfy the requirements of the previous phase. Analysis is the process of checking that the products of each development phase are internally consistent and properly apply the chosen techniques. Software tool guidance associated with the verification and analysis phase covers verification testing, tool assessment methods, and documentation of tool use. Guidance for verification testing states that special test tools (e.g., fault injection tools, in-circuit emulators) might be required to test exception conditions (e.g., divide by zero, out of range) and personnel who are planning and conducting the tests should be given appropriate training in the use of test tools, procedures, and techniques. These personnel should be independent from the development team. Software tool guidance associated with tool assessments defines tool categories and provides tool requirements for each of those categories. Specifically, the guidance states that tools employed in producing software belong to one of the two following broad categories:

- (1) Software development tools, whose output (possibly transformed) becomes part of the program implementation and which can therefore introduce errors (e.g., code generators, compilers, and linkers);
- (2) Software verification tools, which cannot introduce errors but may fail to detect them (e.g., static analysis tools and test coverage monitors).

Further guidance states that the functionality of tools in either category should be precisely defined and all tools should have sufficient safety dependability to ensure that the tool does not jeopardize the safety of the end product. Software development tools should have a precisely defined domain of applicability and should be of the highest dependability level if the development tool output is used without further review. The dependability of development tools may be relaxed if its output is subjected to verification or if reverse engineering is used. Analysis and checks performed by software verification tools should be well defined and the safety dependability may also be relaxed because verification output will be closely scrutinized.

Final software tool guidance associated with the verification and analysis phase states that preparation for verification should be documented at the beginning of each phase and that documentation should include tool assessment criteria. Further guidance states that verification tools used during a verification process should be completely identified (including version and configuration) and any input or set-up parameters used should be recorded.

An ISL observation, based on our analysis of Section 10 of IAEA NS-G-1.1, is that the safety guide is not clear as to whether the two broad categories of tools are meant to address all types of tools used to develop software including, for example, CM tools and requirements specification tools. The qualification requirements for these types of tools should be less rigorous than compilers, linkers, and verification tools. Therefore, IAEA NS-G-1.1 should expand the tool categories and tailor the requirements for each category of tools.

Section 11 of IAEA NS-G-1.1 contains guidance for the computer system integration phase. Computer system integration consists of loading software into the hardware system, testing that

the software-hardware interface requirements are satisfied, and testing that all the software can operate in the integrated software-hardware environment. The only software tool guidance associated with computer system integration is that tools supporting the verification of system integration should be considered.

Sections 12 through 14 of IAEA NS-G-1.1 contain guidance for the life cycle phases of computer system validation, installation and commissioning, and operation. No guidance for the use, verification, validation, review, or approval of software tools is provided for these life cycle phases.

Section 15 of IAEA NS-G-1.1 contains guidance for post-delivery modification of the computer system. Software tool guidance associated with the post-delivery modification phase states that adequate change control procedures for tools that impact plant safety issues should be in place so that independent experts can assess the adequacy of the proposed tool modification and its implementation.

2.8.1.2 IAEA DS-431

IAEA DS-431 is a new safety guide currently in draft form in the comment phase at the time of this report. This new safety standard will supersede IAEA NS-G-1.1 and IAEA NS-G-1.3 and accounts for developments in I&C systems since the publication of those two documents in 2000 and 2002. The standard gives recommendations on the design of I&C systems important to safety. The standard provides high-level guidance on the overall I&C architecture and on the I&C systems important to safety in nuclear power plants for the satisfaction of the safety goals of the plant and is intended to be used in conjunction with an appropriate set of more detailed standards. I&C systems important to safety are part of a safety group and whose malfunction or failure could lead to radiation exposure of site personnel or members of the public such as:

- (1) Reactor protection systems
- (2) Reactor control, reactivity control, and their monitoring systems
- (3) Systems to monitor and control reactor cooling
- (4) Systems to monitor and control emergency power supplies
- (5) Systems to monitor and control containment isolation
- (6) Accident monitoring instrumentation
- (7) Effluent monitoring systems
- (8) I&C for fuel handling

Section 2 of IAEA DS-431 provides guidance for the application of management system standards and life cycle processes for the development of I&C. Management systems include the organizational structure, culture, policies, resources, and processes for developing an I&C system that meets safety requirements. With respect to software tools, the management

process should provide for the control of documents, products (including tools), and records. Section 2 of IAEA DS-431 also gives guidance on generic processes for I&C design and implementation of specific I&C development activities. Confidence in the correctness of modern I&C systems derives more from the discipline of the development process; therefore, many industries have applied development processes that are commonly represented as life cycle models that describe the activities for the development of electronic systems and the relationships between these activities.

Section 2 of IAEA DS-431 also provides guidance for the use of life cycle models and specific guidance for process planning, coordination with human factors, and computer security activities. Process planning guidance states that before initiating any technical activity, a plan identifying the inputs, products, and processes, of that activity should be prepared and approved in accordance with the management system. The only guidance relevant to tools in the process planning phase is that plans for the development of I&C systems need to address the qualification and use of tools. No guidance is provided for tools used in the coordination with human factors or for computer security activities.

Section 2 of IAEA DS-431 also provides guidance for activities common to all life cycle phases including CM, hazards analysis, verification, validation, probabilistic safety analysis, safety assessment, and documentation. Configuration management covers the control of documents, review and approval of changes to documents, controls on verification activities, and control and recording of products. With respect to tools, CM should include techniques and procedures for establishing and maintaining a chronological record of tools used at a particular point in the design process. Also, all I&C items including software tools that are used to produce, control, configure, verify, or validate I&C components and the tool parameter settings should be designated, given a unique identification, and placed under configuration control. No tool guidance is provided for hazards analysis, verification, validation, probabilistic safety analysis, safety assessment, or documentation activities.

Section 2 of IAEA DS-431 also discusses guidance for the specific life cycle activities of requirements specification, selection of pre-developed items, system design implementation, system integration, system validation, installation, overall I&C integration, commissioning, operation, maintenance, and modifications. Section 2 of IAEA DS-431 acknowledges that software tools are typically used to control the issue of modules for assembly into system components and to control the build used for system validation and for on-site operation; however, no guidance is provided. The only software tool guidance relates to modification activities. When software tools are modified or upgraded, the level of rigor applied in justifying and executing the change should be pre-determined and based upon the affected systems' role and function in ensuring the safety of the nuclear power plant, in association with the existing systems that will remain in operation after the work. Also, the consequences of a tool update or change between the time of initial development and modification may be significant and should be subject to impact assessment.

Section 3 of IAEA DS-431 provides guidance for the identification of I&C functions and content of I&C design bases. No software tool guidance is provided in this section.

Section 4 of IAEA DS-431 provides guidance on the I&C architectural design, content of the overall I&C architecture, content of the individual I&C system architectures, independence, and consideration of CCF and diversity. No software tool guidance is provided in this section other than acknowledging that CCFs might happen because of errors in development tools.

Section 5 of IAEA DS-431 describes the safety classification scheme that is used to grade the recommendations of this guide according to the safety significance of items to which they apply. This section does not contain software tool guidance and only provides general guidance for the safety classification process.

Section 6 of IAEA DS-431 provides general recommendations applicable to the overall characteristics and design activities of I&C systems important to safety including designing for reliability, qualifying equipment, coping with aging and obsolescence, controlling access to systems important to safety, testing and testability during operation, maintainability, removing systems from service for testing or maintenance, determining setpoints, and marking and identifying items important to safety. This section does not contain software tool guidance.

Section 7 of IAEA DS-431 provides guidance specific to certain systems such as the reactor protection system, certain types of equipment such as sensors, and to certain technologies such as digital systems and integrated circuits configured with hardware description languages (HDL). The software tool guidance pertaining to HDL programmed devices (HPD) states that only standardized HDLs with qualified and compatible tools should be selected for programming the HPD and that the design of the HPD should ensure that the behavior is deterministic by using an internal synchronous design which favors correctness and testability and allows for the best use of design and verification tools.

Section 7 of IAEA DS-431 also discusses the benefits of software tools and provides examples of software tools. The benefits of tools include reducing the risk of introducing faults during I&C development; improving the probability that faults will be found during checking, verification, and validation; increasing the integrity of the I&C development process and hence component reliability; and economic benefits by reducing the time and human effort required to produce systems, components, and software. Examples of tools include requirements management tools, transformational tools (e.g., code generators, compilers, tools that transform text or diagrams to a lower level of abstraction); CM and control tools, V&V tools (e.g., static code analyzers, test coverage monitors, plan simulators), and security testing tools.

Section 7 of IAEA DS-431 also discusses high-level tool guidance including:

- (1) Tools should be used to support all aspects of the I&C development life cycle where benefits result through their use and where tools are available; however, the benefits and risk of using a tool should be balanced against the benefits and risk of not using a tool.
- (2) Choose tools that limit the opportunity for making errors and introducing faults, but maximize the opportunity for avoiding or detecting faults.

-
- (3) Tools should be selected that will remain available throughout the system's service life and be compatible with other tools used during system development.
 - (4) The functionality and limits of applicability of all tools should be identified and documented and the tools and their output should not be used outside their declared functionality or limits of application without prior justification.
 - (5) Tools should be verified and assessed consistent with the tool reliability requirements, the type of tool, the potential of the tool to introduce fault or fail to make the user aware of existing faults, and the extent to which the tool may affect redundant elements of a system or diverse systems. Examples of situations that can affect the degree of verification and assessment needed include, for example:
 - (a) tools that have the ability to introduce faults need to be verified to a greater degree than tools that are demonstrated to not have that capability;
 - (b) tools that can fail to make the user aware of existing faults need to be verified to a greater degree than tools that do not have that capability;
 - (c) verification is not necessary for tools when the output of the tool is systematically and independently verified; and
 - (d) less rigor in tool verification may be accepted if there is mitigation of any potential tool faults (e.g. by process diversity or system design).
 - (6) The verification and assessment of software tools should take into account experience from prior use, including experience of the developers and experience gained from the processes in which the tools are used.
 - (7) The choice, verification, and assessment of tools should be justified and documented.
 - (8) All tools should be under appropriate CM.
 - (9) Tool settings used during the development, verification, or validation of baseline equipment, software, or HDL configured devices should be recorded in the development records to ensure final software consistency and help in assessing the origin of a fault which might lie in the source code, in the tool, or in the tool settings. Information about the tool settings used may be critical to assessing the potential for CCFs due to software tools.

Section 8 of IAEA DS-431 discusses human-machine interface considerations including control rooms, accident monitoring, operator communications systems, general human factors

engineering principles, and recording of historical data. This section contains no guidance for software tools.

Section 9 of IAEA DS-431 gives guidance on the development of software for computer-based I&C systems important to safety. Guidance is provided for the life cycle processes and activities associated with software requirements, design, implementation, verification and analysis, pre-developed software, and third party assessment. With respect to software requirements, guidance is provided that states the software requirements specification should allow for the capabilities of the computers, tools, and similar existing systems to ensure that the software requirements are feasible. Software design guidance states that software design documentation should include design phase implementation constraints (i.e., any need to ensure diversity, and particular attributes of the programming languages, compilers, subroutine libraries, and other supporting tools) that should be justified or be traceable to higher-level requirements or constraints. Software implementation guidance states that the software implementation should be established using a predetermined combination of techniques commensurate with the system's importance to safety, covering languages, tools, coding practices, analysis, review, and testing. Also, a suitable set of implementation tools should be selected with the aim of minimizing error. Software verification and analysis guidance states that a software verification plan should document the functionality of any verification tool, including expectations and limitations on how it is to be used.

2.8.2 Summary

IAEA safety guides contain a few high-level software tool guidelines that could form a good technical basis for developing new regulatory guidance for the commercial nuclear power industry. IAEA NS-G-1.1 focuses on documentation requirements while IAEA DS-431 provides high-level guidance on the overall I&C architecture and on the I&C systems important to safety in nuclear power plants for the satisfaction of the safety goals of the plant. The IAEA safety guides do not discuss software classifications and lack tool qualification, review, and approval guidance.

IAEA NS-G-1.1 is a safety guide that provides guidance for demonstrating the safety of software used in nuclear power plant computer-based systems including guidance for software tools used in the computer-based system and software development life cycle. IAEA NS-G-1.1 states that only well-established tools should be used and documented in the software development plan, QA program description, V&V plan, and CM plan. Tools used should be mutually compatible and qualified to a level commensurate with their functionality. V&V should be done using different techniques and tools from those used to develop the software to ensure technical independence and diversity. IAEA NS-G-1.1 recognizes that the executable code can be produced from the system specifications using code generating tools that translate a software requirements specification into software design requirements, software design requirements into source code, and source code into executable code. IAEA NS-G-1.1 states that these types of translators should be thoroughly tested and if not thoroughly tested, additional analysis and verification, manual or using another tool, should demonstrate that the translator is correct.

IAEA NS-G-1.1 defines two tool categories (i.e., software development tools and software verification tools) and provides tool requirements for each of those categories. IAEA NS-G-1.1 states that the functionality of tools in either category should be precisely defined and all tools should have sufficient safety dependability to ensure that the tool does not jeopardize the safety of the end product. Software development tools should be of the highest dependability level; however, the dependability may be relaxed if its output is subjected to verification or if reverse engineering is used. Analysis and checks performed by software verification tools should be well defined and the safety dependability may also be relaxed because verification output will be closely scrutinized. An ISL observation, based on an analysis of IAEA NS-G-1.1, is that IAEA NS-G-1.1 should expand the tool categories and tailor the requirements for each category of tools because not all software development tools or software verification tools are equally important. For example, the safety dependability of verification tools that automatically modify source code should be higher than verification tools that do not have that type of functionality. Also, CM tools or tools that transform high-level system requirements into low-level design requirements or source code should have a safety dependability level lower than software tools that are involved in the compilation process.

IAEA DS-431 is a new draft safety guide that will supersede IAEA NS-G-1.1 and IAEA NS-G-1.3 and accounts for the latest developments in I&C systems. IAEA DS-431 gives high-level guidance on the overall I&C architecture and on the I&C systems important to safety in nuclear power plants for the satisfaction of the safety goals of the plant. IAEA DS-431 states that plans for the development of I&C systems need to address the qualification and use of tools. IAEA DS-431 states that CM should include techniques and procedures for establishing and maintaining a chronological record of tools used at a particular point in the design process and all software tools that are used to produce, control, configure, verify, or validate I&C components and the tool parameter settings should be designated, given a unique identification, and placed under configuration control.

IAEA DS-431 discusses the benefits of software tools and acknowledges the use of several different types of software tools including requirements management tools, transformational tools (e.g., code generators, compilers, tools that transform text or diagrams to a lower level of abstraction); CM and control tools, V&V tools (e.g., static code analyzers, test coverage monitors, plan simulators), and security testing tools. IAEA DS-431, by identifying these types of tools, has abandoned the two broad tool categories of IAEA NS-G-1.1. IAEA DS-431 provides guidance for the selection of software tools including balancing the benefits and risk of using or not using a tool. IAEA DS-431 states that tools should be verified and assessed consistent with the tool reliability requirements, the type of tool, the potential of the tool to introduce a fault or fail to make the user aware of existing faults, and the extent to which the tool may affect redundant elements of a system or diverse systems. Tools that can introduce faults or fail to detect faults need to be verified to a greater extent than other tools; however, verification is not necessary if the tool output is systematically and independently verified. An ISL observation, based on our analysis of IAEA DS-431, is that the guidance is high-level and does not tailor the guidance for different types of tools. The high-level guidance of IAEA DS-431 needs to be supplemented with low-level requirements that tailor the requirements to different types of tools.

IAEA DS-431 and IAEA NS-G-1.1 include guidance for software tools used in a computer-based system and software development life cycle. Based on the analysis of IAEA DS-431 and IAEA NS-G-1.1, the following essential elements of the documents relative to software tools have been determined.

- (1) The diversity of methods, languages, tools, and personnel should be taken into consideration along with the use of diverse functions and system components at different levels of the design.
- (2) All tools used should be mutually compatible and qualified to a level commensurate with their function in the development of the software and in the safety demonstration.
- (3) The techniques used to gain confidence in the tools should be specified and documented.
- (4) Well established methods, languages, and tools for software development should be used and methods, languages, and tools that are still at the research stage should not be used.
- (5) Tools to be used in the software development process should be identified in the development plan; selected to facilitate the proper application of the methods, standards, and procedures selected; and planned in advance for the entire project to help in the selection of an integrated set of tools.
- (6) Tools should be properly qualified for their function in the development, management, or verification of the system and that the correctness of their output should be ensured by a tool certification process, by means of cross-verification, or by reverse engineering.
- (7) A QA plan should be prepared that covers external suppliers and should identify the tools to be used on the project and the procedures for qualifying the tools.
- (8) V&V should be done using different techniques and tools to maintain technical independence.
- (9) All items of software development, such as compilers, development tools, configuration files, and operating systems, should be under CM control.
- (10) Tools used to assist the validation of the completeness and consistency of the computer system requirements should be validated to a level of confidence equivalent to the level required from the design process.
- (11) If parts of the computer system design or code are generated by tools, then a separate software requirement document may be unnecessary for those parts

if the computer system requirements are sufficiently detailed and formally documented.

- (12) Code can be produced from the system specifications using combinations of two distinct approaches (i.e., the classical development process or the use of code generating tools). The choice between these two approaches depends on the tools and resources available to the project and should take into account trade-offs between design and the demonstration of the dependability of tools.
- (13) Only validated tools should be used.
- (14) A suitable set of tools should be selected.
- (15) Translators should be thoroughly tested and available throughout the system lifetime. If the translator used is not thoroughly tested, additional analysis and verification, manual or using another tool, should demonstrate that the translator is correct.
- (16) Tools used in the implementation phase should be compatible with tools used in other phases of software development.
- (17) Tools employed in producing software belong to one of two broad categories: software development tools and software verification tools.
- (18) Software development tools should have a precisely defined domain of applicability and should be of the highest dependability level if the development tool output is used without further review. The dependability of development tools may be relaxed if its output is subjected to verification or if reverse engineering is used.
- (19) Analysis and checks performed by software verification tools should be well defined and the safety dependability may be relaxed because verification output will be closely scrutinized.
- (20) All I&C items including software tools that are used to produce, control, configure, verify, or validate I&C components and the tool parameter settings should be designated, given a unique identification, and placed under configuration control.
- (21) When software tools are modified or upgraded, the level of rigor applied in justifying and executing the change should be pre-determined and based upon the affected systems' role and function in ensuring the safety of the nuclear power plant, in association with the existing systems that will remain in operation after the work.

- (22) The consequences of a tool update or change between the time of initial development and modification may be significant and should be subject to impact assessment.
- (23) The benefits and risk of using a tool should be balanced against the benefits and risk of not using a tool.
- (24) Choose tools that limit the opportunity for making errors and introducing faults, but maximize the opportunity for avoiding or detecting faults.
- (25) Tools should be selected that will remain available throughout the system's service life and be compatible with other tools used during system development.
- (26) The functionality and limits of applicability of all tools should be identified and documented and the tools and their output should not be used outside their declared functionality or limits of application without prior justification.
- (27) Tools should be verified and assessed consistent with the tool reliability requirements, the type of tool, the potential of the tool to introduce a fault or fail to make the user aware of existing faults, and the extent to which the tool may affect redundant elements of a system or diverse systems.
- (28) Verification is not necessary for tools when the output of the tool is systematically and independently verified.

Based on the analysis of IAEA DS-431 and IAEA NS-G-1.1, ISL has made the following important observations and recommendations.

- (1) IAEA NS-G-1.1 should expand the two tool categories (i.e., development and verification) and tailor the requirements for each category of tools.
- (2) The IAEA safety guides do not discuss software classifications and lack tool qualification, review, and approval guidance.
- (3) The IAEA DS-431 guidance for software tools is high-level and does not tailor the guidance for different types of tools.
- (4) The high-level guidance of IAEA DS-431 needs to be supplemented with low-level requirements that tailor the requirements to different types of tools.

2.9 Electric Power Research Institute Analysis

The Electric Power Research Institute (EPRI) was formed in 1972 as an independent research and development organization to support the electricity industry in generating, delivering, and using electricity that is affordable, reliable, and environmentally responsible. EPRI conducts research, development, and demonstration for the benefit of the public by bringing together

scientists, engineers, and experts from academia and industry to provide short- and long-term solutions in every area of electricity generation, delivery, and use.

2.9.1 Document Analysis

EPRI produces thousands of products annually, ranging from white papers and newsletters to comprehensive reports. Six reports are relevant to software tools; however, only five of the reports will be analyzed in the report. EPRI NP-5652 and EPRI TR-102260 discuss utilization of CGIs and commercial-grade dedication of equipment for nuclear safety-related applications. Although these two documents are primarily focused on hardware components, the recommended approach to commercial-grade dedication applies to software tools. EPRI TR-106439 and TR-107339 provide guidance for evaluating and accepting commercial-grade digital equipment for nuclear safety-related applications. EPRI TR-107330 provides requirements for qualifying commercially available programmable logic controllers for nuclear safety-related applications. Although EPRI TR-107330 discusses software tools, the discussion is limited to software tool requirements specific to the programming, debugging, and documenting of a programmable logic controller application. Therefore, an analysis of EPRI TR-107330 will not be discussed in this report. The sixth report, EPRI TR-1025243, Revision 1, issued in December 2013, is an updated version of the initial report and provides guidance for commercial-grade design and dedication of software used in nuclear safety-related applications.

2.9.1.1 EPRI NP-5652

EPRI NP-5652 provides guidance for procuring and using a CGI in safety-related nuclear power plant applications. The decreasing availability of nuclear plant qualified replacement parts and components designed and manufactured in accordance with the 10 CFR Part 50, Appendix B, QA program has forced utilities to purchase parts and components as commercial grade and dedicate the parts for safety-related use. EPRI NP-5652 contains a generic process for the acceptance of CGIs for use in a nuclear safety-related application. The generic process is primarily for hardware components and parts; however, the process could be adapted to software tools.

The generic process for the acceptance of CGIs for installation in nuclear safety-related applications consists of the following six steps.

1. Safety Function – The safety function of a CGI should be determined by analysis. The analysis should consider the function of the item and the safety classification of the parent component. The safety function of the item should also consider the ways in which the item could fail and whether those failures would prevent the parent component or associated components from performing their safety function. Items which are determined to be nonsafety-related need not be dedicated. Items which are determined to be safety-related can be procured as a basic component if it was designed and manufactured in accordance with the 10 CFR Part 50, Appendix B, QA program or the item can be procured as a CGI and accepted through the commercial grade dedication process.

An ISL observation, based on our analysis of this process step, is that the step can be applied to software tools. The safety function of the software tool can be determined based on the classification of the computer-based component being developed with the tool. Nonsafety-related software tools would not need to be dedicated consistent with their hardware counterparts.

2. Commercial Grade Confirmation – The item should be confirmed as a CGI by determining whether the item meets the criteria set forth in the CGI definition. An item is a CGI if the item is not subject to design or specification requirements unique to nuclear facilities, is used in applications other than nuclear facilities, and is ordered from the manufacturer or supplier based on specifications published in a catalog or product description. If the item is not a CGI, the item must be procured as a basic component.

An ISL observation, based on our analysis of this process step, is that the step can be applied to software tools. A software tool can be confirmed as a CGI if the tool has not been developed in accordance with a formal and recognized software life cycle development process, is used in non-nuclear computer-based applications, and is considered off-the-shelf and purchased based on a catalog description.

Note that in 1995, the NRC revised the definition of a CGI in 10 CFR 21.3 for nuclear power plants licensed under 10 CFR Part 50, to state that CGIs for this type of facility are items intended for use as basic components, but that were not designed and/or manufactured under a 10 CFR Part 50, Appendix B, QA program. Thus for NPPs, CGI's did not have to meet the old definition stated above, which is still applicable to other types of facilities subject to Part 21. For NPPs, the requirement for an item to be eligible for dedication is that all its critical characteristics can be verified.

3. Critical Characteristics – The critical characteristics of the CGI must be identified so they can be verified to provide reasonable assurance that the item received is the item specified. An item's critical characteristics can include a part number, identification markings, or performance characteristics. Verification of the CGI's critical characteristics provides assurance of acceptability.

An ISL observation, based on our analysis of this process step, is that the step can be applied to software tools. The critical characteristics of a software tool can be determined in part from its version number, operational environment, functional requirements, and the output produced by the tool.

Note that the NRC definition of critical characteristics, first promulgated in GL 91-05 and later codified in 10 CFR 21.3 states that critical characteristics, when verified, provide reasonable assurance that the item will perform its intended safety function.

4. Acceptance Method – Acceptance of a CGI can be based on one or more of the following four alternative methods. These four methods ensure that a CGI which is received meets the requirements of the item specified. The applicability of each of these four methods to software tools is discussed below.

- a. Method 1 – Special Tests and Inspections. Method 1 consists of special tests and inspections conducted during standard receipt inspections or after receipt to verify selected critical characteristics. These tests may include post-installation tests. Method 1 should be used when the purchaser desires to verify critical characteristics after the item is received because the items are simple, obtained from multiple suppliers, or conducive to post-installation testing. Once the critical characteristics are verified, the item may be accepted for use in safety-related applications with supporting documentation of the test results.

An ISL observation, based on our analysis of Method 1, is that the method may be suitable for software tools. Physical inspections of software tool executables analogous to hardware part numbers or identification markings would include a verification of the tool version, installation package, intended operational environment, development environment, and other critical software development characteristics. Pre-installation tests are possible by installing the tool on a standalone compute platform before installing the tool on its designated final platform. Post-installation testing of a software tool would be the final and most effective acceptance method by confirming that the critical characteristics required of the software tool are present in the output of the test suite.

- b. Method 2 – Commercial Grade Survey of Supplier. Method 2 consists of a commercial-grade survey of the supplier. The survey allows the purchaser to take credit for the commercial controls used by the supplier on a CGI. Method 2 should be used when the supplier's QA program, procedures, and practices are acceptable to the purchaser and critical characteristics of the CGI cannot be verified by inspections or tests.

An ISL observation, based on our analysis of Method 2, is that the method may be applied to software tools. A survey of the software tool developer's life cycle development process including the supporting QA, V&V, and configuration control processes would satisfy the intent of this method.

Note that NRC GL 89-02 promulgated the staff position that Method 2 should not be employed as the basis for accepting items from suppliers with undocumented commercial quality control programs or with programs that do not effectively implement their own necessary controls. It also states that the commercial-grade survey should be item- and critical characteristic-specific, that the survey confirms that the supplier

verifies all critical characteristics, and that the supplier has implemented an effective commercial QA and quality control program.

- c. Method 3 – Source Verification. This method accepts a CGI based on a source verification of a supplier. In the traditional source verification, the purchaser or its agent personally verifies at the factory the critical characteristics of the actual item(s) being purchased or directly observes the supplier's QA activities prior to shipment of the item. Method 2 is intended to qualify a supplier's overall commercial grade quality control activities, while Method 3 is better suited for items procured on an infrequent or expedited basis. This method may require the purchaser's specification of hold points to enable the source verifier's direct verification or observation of the supplier's verification of critical characteristics, but it allows for independent purchaser verification of critical characteristics that may not be readily verifiable in the finished product without reliance on the supplier's routine process.

An ISL observation, based on our analysis of Method 3, is that the method could be applied to software tools. For software tools, a source verification could include an agent or purchaser's verification of the requirements specification, design requirements, and code requirements. A source verification for software tools could also include direct observation of QA audits and reviews.

- d. Method 4 – Acceptable Supplier/Item Performance Record. Method 4 allows the acceptance of a CGI based on the confidence in the item achieved through proven performance of the item. Proven performance can be based on historical verification of critical characteristics using one of the other three methods of acceptance or based on industry-wide performance data. Method 4 is best suited for situations in which the performance of the item is monitored and the historical performance of the item is well documented.

An ISL observation, based on our analysis of Method 4, is that this method would be the most reliable method of acceptance for COTS software tools. An assessment of the historical performance of a software tool can be performed by examination of bug reports and of documentation of successful use of the tool in other applications and industries.

Note that NRC GL 89-02 promulgated the staff position that Method 4 should not be employed alone unless (a) the established historical record is based on industry-wide performance data that are directly applicable to the CGI's critical characteristics and the intended safety-related application; and (b) the manufacturer's measures for the control of design,

process, and material changes have been adequately implemented as verified by audit (multi-licensee team audits are acceptable).

5. Acceptance Delegation – The organization responsible for accepting a CGI for safety-related use must include the acceptance process within its 10 CFR Part 50, Appendix B, QA program.

An ISL observation, based on our analysis of this process step, is that the step can be applied to software tools. The acceptance of a software tool should be thoroughly documented as part of the software development life cycle process associated with tool use.

6. Reportability and Traceability – The organization responsible for accepting a CGI for safety-related use should control the reportability and traceability of the item along with other safety-related items.

An ISL observation, based on our analysis of this process step, is that the step can be applied to software tools. The reportability and traceability of a software tool should be thoroughly documented as part of the software development life cycle process associated with tool use.

With respect to the critical characteristic process, NRC GL 91-05 took exception to part of the EPRI definition of critical characteristics (i.e., that critical characteristics, when verified, provide reasonable assurance that the item received is the item specified). The NRC position was that part number verification alone was not sufficient and that critical characteristics, all of which must somehow be verified, provide reasonable assurance that the item will perform its intended safety function. This position was codified in the 1995 revision to 10 CFR Part 21.

The essential elements of this generic acceptance process are the four acceptance methods. The four acceptance methods were conditionally endorsed by the NRC in GL 89-02, with restrictions on the use of Methods 2 and 4. As stated above, NRC GL 89-02 states that Method 2 should not be employed as the basis for accepting items from suppliers with undocumented commercial quality control programs or with programs that do not effectively implement their own necessary controls. It also states that the commercial-grade survey should be item- and critical-characteristic-specific, that the survey confirms that the supplier verifies all critical characteristics, and that the supplier has implemented an effective commercial QA and quality control program.

NRC GL 89-02 also states that Method 4 should not be employed alone unless (a) the established historical record is based on industry-wide performance data that is directly applicable to the item's critical characteristics and the intended safety-related application; and (b) the manufacturer's measures for the control of design, process and material changes have been adequately implemented as verified by audit. Note that control of design and material changes for hardware corresponds roughly to SCM.

Further clarification of the NRC staff position on this guidance was promulgated by the NRC in GL 91-05 as stated above. In 1995, the NRC's new definitions of basic component, CGI,

dedication, and critical characteristic were codified in a revised 10 CFR Part 21 along with other new dedication-related requirements (i.e., that the dedicating entity is responsible for reporting and maintaining auditable records of dedications, and that the dedication must be performed under the applicable criteria of 10 CFR, Part 50, Appendix B). Note that the overall intent of a dedication process is to meet the requirements, primarily of Appendix B, Criterion III, "Design Control" (in particular, the review for suitability of application and design verification), and of Criterion VII, "Control of Purchased Material, Equipment, and Services" (in particular, qualification of suppliers and examination of products upon delivery).

The EPRI commercial-grade dedication guidance defines two phases for the dedication process: technical evaluation and acceptance. In the technical evaluation phase, the dedication plan is developed to include determination of the item's safety functions, other application-specific requirements including ambient and process design-basis service conditions, the known vulnerabilities and deficiencies of the item based on industry operating experience, and applicable regulatory requirements and guidance. From this information, the critical characteristics are derived, and plant application-relevant verification methods and acceptance criteria are specified. The NRC's position has been that as part of the technical evaluation phase of the dedication process, one or more of the four most appropriate acceptance methods should be selected for verification of each critical characteristic along with the most plant application-relevant verification procedure (e.g., type of test or inspection), and the most application-relevant acceptance criteria. In the acceptance phase, one or more of the four acceptance methods should be used to execute the dedication plan and actually perform the verification of the critical characteristics. This generic process was written for applicability to hardware components and parts; however, the process could be applied to software tools with minor clarifications of the methods of acceptance.

2.9.1.2 EPRI TR-102260

EPRI TR-102260 contains supplemental guidance for dedication of CGIs for nuclear safety-related applications to address industry and NRC actions since the original issuance of EPRI NP-5652.

As part of the clarification of NP-5652, EPRI TR-102260 provides a list of requirements from American National Standards Institute (ANSI) Standard N18.7, paragraph 5.2.13. One of these requirements states:

"In those cases where the QA requirements of the original item cannot be determined, an engineering evaluation shall be conducted by qualified individuals to establish the requirements and controls. This evaluation shall assure that interfaces, interchangeability, safety, fit, and function are not adversely affected or contrary to applicable regulatory or code requirements."

Guidance in EPRI NP-5652 and TR-102260 are intended to comply with this requirement. Also with respect to this requirement, NRC GL 91-05 states that in addition to form, fit, and function, manufacturing process and materials should also be considered along with a verifiable reconciliation of changes. An ISL observation, based on our analysis of this requirement and

the statements in NRC GL 91-05, is that the requirement will be instrumental in forming the basis for commercial-grade dedication of software tools where the vendor either cannot or will not provide documentation of the QA requirements. Another ISL observation is that the tool developer's process should be evaluated and that the GL 91-05 term "materials" is analogous to software and software tools. Also, control of design and material changes would be analogous to SCM.

A fundamental concept of commercial-grade dedication discussed in EPRI TR-102260 is in the application of a graded approach that is consistent with 10 CFR Part 50, Appendix B, Criterion II, which states that:

"The quality assurance program shall provide control over activities affecting the quality of the identified structures, systems, and components, to an extent consistent with their importance to safety".

The use of engineering judgment and the concept of reasonable assurance in EPRI NP-5652 and EPRI TR-102260 are accepted techniques in the application of the graded approach consistent with the importance of the item to safety. An ISL observation, based on our analysis of this fundamental concept, is that the graded approach should be extended to the functional safety classification of CGIs. Based on guidance in EPRI NP-5652, the safety classification of a component is determined from an analysis of its safety function and is categorized as either safety-related or nonsafety-related. A graded approach would still be consistent with EPRI NP-5652 in that all safety-related components would need to be dedicated; however, the graded approach would subdivide the classification of safety-related into multiple classifications where the most important safety-related items are subjected to a more rigorous dedication process than less important safety-related items. The graded approach also would still be consistent with the guidance in EPRI NP-5652 in that it would not require nonsafety-related items to be dedicated.

EPRI TR-102260 provides supplemental, detailed guidance on the concept of reasonable assurance as expressed in EPRI NP-5652 with respect to verifying critical characteristics of a CGI to assure that the item received is the item specified. Although not defined in EPRI NP-5652, EPRI TR-102260 describes the term "reasonable assurance" as the level of confidence attained during the acceptance process that the CGI received meets specified requirements. EPRI TR-102260 further describes the term "reasonable" as a level of confidence that is justifiable but not absolute. Reasonable assurance of performance must be based on facts, actions, or observations. However, the decision that reasonable assurance has been attained is inherently subjective and the judgment of reasonability may vary between different observers. Maintaining a methodology based on the guidance of EPRI TR-102260 may help to reduce the subjective nature of the judgment. In clarification of the statement "item received is the item specified," EPRI TR-102260 states that the term "item specified" should be viewed in a broad, generic sense and should not be literally taken to mean that each and every technical requirement of a given item communicated (i.e., specified) in a procurement document needs to be verified to meet 10 CFR Part 50, Appendix B, Criterion VII. There is no direct correlation between specified technical requirements and critical characteristics. Not all technical

requirements correlate to a critical characteristic and critical characteristics can be selected that do not explicitly appear in the technical specification. EPRI TR-102260 cautions that the acceptance process should not be oversimplified by simply checking part or version numbers as a means of acceptance.

EPRI TR-102260 provides supplemental, detailed guidance on the interface between the technical evaluation process and the acceptance process. The technical evaluation process identifies the correct item for an application and is not an integral part of the acceptance of CGIs. Instead, the technical evaluation process supplies the technical information necessary as input to the acceptance process. The acceptance process provides reasonable assurance that the item received meets the specified requirements and only interfaces with the technical evaluation process through the required input to fully implement acceptance. Neither the technical evaluation process nor the acceptance process are substitutes for the design and equipment qualification activities covered by 10 CFR Part 50, Appendix B, Criterion III. The qualification process defines the performance criteria for the equipment specification and in the case where the technical and quality requirements are in existing specifications, EPRI TR-102260 states that a formal technical evaluation process may not be necessary. An ISL observation, based on our analysis of this supplemental guidance, is that the guidance could be improved. Guidance that states a formal technical evaluation may not be necessary is too vague. The guidance should clearly indicate the circumstances under which a technical evaluation is necessary and when it is not. Engineering judgment can be relied upon to establish reasonable assurance that the item received meets specified requirements; however, engineering judgment should not be relied upon to eliminate commercial-grade dedication processes.

EPRI TR-102260 provides supplemental, detailed guidance on the difference between a critical characteristic for design and a critical characteristic for acceptance. Critical characteristics for design are based on an analysis of safety functions supported by a failure mode and effects analysis and include attributes that are essential for the item's functional performance. Critical characteristics for acceptance are based upon the complexity, intended safety-related function, and performance of the CGI. Critical characteristics for acceptance include physical characteristics of an item and are generally a subset of the critical characteristics for design which include both physical characteristics and performance characteristics but not product identification characteristics.

EPRI TR-102260 clarifies that the intent of EPRI NP-5652 was that Method 2, "Commercial Grade Survey of Suppliers," could be used as the sole basis for acceptance of a CGI and that NRC GL 89-02 and 91-05 do not contradict this intent. NRC GL 89-02 merely establishes two guidelines for the use of a commercial grade survey as a basis of accepting a CGI. These two guidelines are:

- (1) Acceptance Method 2, "Commercial-Grade Survey of Supplier," should not be employed as the basis for accepting items from suppliers with undocumented commercial quality control programs or with programs that do not effectively implement their own necessary controls.

- (2) Method 2 should not be employed as the basis for accepting items from distributors unless the survey includes the part manufacturer(s) and the survey confirms adequate controls by both the distributor and the part manufacturer(s).

EPRI TR-102260 thus concludes that when the implementation of Method 2 demonstrates that the supplier documents their quality control program, effectively implements the quality controls, and the commercial-grade survey confirms adequate quality control by both the distributor and manufacturer, then Method 2 can be a sole basis of acceptance coupled with standard receipt inspection.

Note that in its clarification of NP-5652 above, TR-202260 did not mention that GL 89-02 also stated that in using Method 2, the commercial-grade survey should be item- and critical characteristic-specific and that the survey should confirm that the supplier verifies all critical characteristics.

EPRI TR-102260 also contains additional guidance on the use of EPRI NP-5652, Method 4, “Acceptable Supplier/Item Performance Record”. The original intent of Method 4 was to use it as the sole basis of acceptance of a CGI based on a documented performance record of both the item and the supplier of the item. Method 4 eliminates the costly and time consuming activities of Methods 1, 2, and 3 which may not add to the quality or performance of the CGI. EPRI TR-102260 clarifies that Method 4 is appropriate only under certain conditions that depend on item complexity and documentation of supplier/item performance. An ISL observation, based on our analysis of Method 4, is that since any software tool is a complex piece of software, the use of performance records to accept complex software should be done with extreme caution. Additionally, EPRI TR-102260 recommends extreme caution if adequate documentation of the item’s performance cannot be produced. See also the GL 89-02 restrictions on Method 4 discussed above.

An overall ISL observation, based on our analysis of EPRI TR-102260, is that the document only contains supplemental guidance for EPRI NP-5652 with a focus on materials and equipment. No specific discussion is provided for digital equipment, software, or software tools. However, the commercial-grade dedication process can be applied to software and software tools with minor clarifications of the process.

2.9.1.3 *EPRI TR-106439*

EPRI TR-106439 provides guidance on the evaluation and acceptance of commercial-grade digital equipment as part of the commercial-grade dedication process for nuclear safety applications. The guidance helps utilities evaluate, design, and implement upgrades to DI&C equipment and facilitates obtaining regulatory approval for I&C upgrades. EPRI TR-106439 relies on the existing commercial-grade dedication process described in EPRI NP-5652 and EPRI TR-102260 with supplemental guidance to address digital-specific issues. EPRI TR-106439 emphasizes identification of critical characteristics with subsequent verification through testing, analysis, vendor assessments, and review of DI&C operating experience. EPRI TR-106439 supports a graded approach to QA based on the importance of the digital equipment to safety. EPRI TR-106439 identifies several issues associated with the use of commercial-grade

digital equipment in nuclear safety applications. The issues include the potential for CCF resulting from software errors, the effects of electromagnetic interference, and the use and control of equipment (i.e., tools) for configuring computer-based systems.

As stated in 10 CFR Part 21, the goal of commercial-grade dedication is to provide reasonable assurance that an item will perform its safety function. It also states that a properly dedicated CGI is deemed to be equivalent to an item designed and manufactured in accordance with a 10 CFR Part 50, Appendix B, QA program. EPRI TR-106439 states that the key elements of the dedication process are a technical evaluation to define the requirements for the device, generating a set of critical characteristics for acceptance from the device requirements, and applying the methods of EPRI NP-5652 to verify the critical characteristics. Since EPRI NP-5652 applies primarily to hardware, EPRI TR-106439 states that new critical characteristics and additional verification activities are needed when the methods of EPRI NP-5652 are applied to digital devices. EPRI TR-106439 also states that no one method from EPRI NP-5652 will suffice by itself and that for many digital devices, a combination of Methods 1, 2, and 4 are needed.

EPRI TR-106439 provides a critical characteristic matrix for digital equipment that lists typical critical characteristics and provides examples of acceptance criteria and verification methods. The matrix covers physical characteristics, performance characteristics, and new critical characteristics for digital devices that are related to dependability (i.e., reliability and maintainability). Physical characteristics included in the matrix are part number, firmware version, software revision level, size, type of mounting, power requirements, and data communication characteristics. Performance characteristics included in the matrix are input processing functionality, required algorithms or functions, output signal requirements, test and diagnostic functions, response time, accuracy, parameter ranges, data throughput rate, and environmental conditions required to maintain operability. Dependability characteristics cover digital equipment reliability and maintainability. Reliability is concerned with built-in quality including the quality of design, quality of manufacture, failure management, and compatibility with human operators. Maintainability is concerned with configuration control and traceability. The critical characteristic matrix provides examples of acceptance criteria, methods of verification, and application of the methods for each of the example critical characteristics listed in the table. All four methods of acceptance are covered by the examples in the matrix demonstrating that one or more of the four EPRI NP-5652 acceptance methods should be used to verify each critical characteristic. An ISL observation, based on our analysis of this critical characteristic matrix, is that most of the examples of critical characteristics for digital equipment apply to the digital hardware, some apply to the software, and most would not be applicable to software tools. The examples that would apply to software tools are critical characteristics related to dependability that rely on the review of vendor design, development, and verification processes, review of QA practices, design reviews, failure analysis, review of operating history, review of configuration control practices, and review of documentation to verify the critical characteristics to complete the digital equipment commercial-grade dedication process.

One of the key concepts discussed in EPRI TR-106439 is that of engineering judgment. EPRI TR-106439 states that engineering judgment must be based on quantifiable critical

characteristics and adequate evidence. EPRI TR-106439 also states that the documentation of the judgment should be sufficient to allow a comparably qualified individual to come to the same conclusion. An ISL observation, based on our analysis of the use of engineering judgment, is that judgment should be restricted to determining whether an adequate level of assurance has been attained during the commercial-grade dedication process and for selecting the best EPRI NP-5652 method of verifying the critical characteristics. Engineering judgment should not be used to eliminate or as a substitute for a dedication process (e.g., failure analysis).

Another key concept discussed in EPRI TR-106439 is that of CM. EPRI TR-106439 states that the CGI supplier or manufacturer must have a documented CM program that is consistent with relevant standards and acceptance practices and the program must be strictly adhered to and well documented from initial development through changes and releases. An ISL observation, based on our analysis of this CM guidance, is that it would be difficult to apply this guidance to the dedication of COTS software tools because the original manufacturer may include updates to software tools along with updates to the operating environment (i.e., regular operating system updates). Therefore, it is imperative that these “automatic” updates are accounted for and that as much information about the update as possible be included in the software tool documentation set.

Commercial dedication of software tools is not directly addressed in EPRI TR-106439. Many of the steps described in the dedication process would be difficult to accomplish for software tools. Vendors that produce COTS software tools would likely not be interested in the time and expense of the commercial dedication process given the relatively small market. This is especially true when one considers the number of choices that developers have for compilers, linkers, integrated development environments, and other tools. Because of this, the burden of commercial dedication of software tools will fall to the vendors developing the final software, who will be using the tools, or to the licensee.

2.9.1.4 EPRI TR-107339

EPRI TR-107339 gives more detailed “how to” guidance for evaluating commercial digital equipment for nuclear safety applications and other applications requiring high reliability through tailoring of the guidance in EPRI TR-106439 consistent with the importance of the digital equipment to safety and economics. The primary focus of EPRI TR-107339 is on the differences in the technical evaluation and acceptance process required for digital, software-based equipment and systems. EPRI TR-107339 provides guidance for each of the phases of the CGI dedication process including project definition, defining the detailed requirements, defining the critical characteristics for acceptance, formulation of an acceptance strategy, and verification of critical characteristics.

EPRI TR-107339 provides supplemental guidance for the project definition and planning process. Specifically, supplemental guidance is provided for grading, evaluating complexity, and performing cost-benefit evaluations. The term “grading” refers to the implementation of a graded approach in determining the scope of activities required to commercially dedicate the digital equipment. The term “project definition” refers to all the activities performed prior to committing to the project. The project definition process involves iteration to arrive at a final

design concept. The output of the project definition process includes an established schedule and a narrow, well-defined list of equipment options and suitable vendors. EPRI TR-107339 states that the project definition process involves the following activities:

- (1) Pre-Conceptual Design – This first activity in the project definition process defines the objectives of the digital equipment installation and any increase in functionality planned relative to the existing system. This activity also defines the systems and components to be modified or impacted by the digital equipment installation, defines early design concepts which will fulfill the objectives, and identifies high-risk areas and important system-level failure modes that need to be considered in subsequent activities.
- (2) Identification of Design Basis and Licensing Basis Requirements – This second activity is performed to understand the design basis and licensing requirements for the system and equipment being modified or replaced. This understanding is necessary to assess the safety significance of the equipment to define the graded approach to dedication.
- (3) Initial Grading – The third activity is a multi-step process to provide the necessary input for defining the critical characteristics and the verification methods. The first step of the initial grading activity is identifying the safety significance of the affected plant system or component in which the CGI will be used or which will be affected by the item. After the system safety significance has been identified, the second step is to identify the safety significance of the specific CGI based on the safety function of the equipment as determined by the technical evaluation and initial system-level failure or hazard analysis. The third step is to assess the complexity of the equipment and application based on the information available recognizing that the internal complexity of the CGI is not yet known since a particular device and vendor have not yet been selected. The fourth step is to define the scope and rigor of activities needed for evaluation and acceptance of the CGI to ultimately provide “reasonable assurance” that the item will perform its intended safety function. The defined scope and rigor of activities are then applied in the definition of the critical characteristics for acceptance and the determination of verification methods used to verify the critical characteristics.
- (4) Identification and Screening of Products and Suppliers – The fourth activity consists of identifying the large number of candidate CGIs that are suitable for the application and to screen the items and vendors or suppliers to determine their viability. The viability assessment should consider the vendor or supplier QA practices; the industry standards followed by the vendor or supplier; vendor support for a survey of QA, design, and implementation processes; CM practices; and availability of documentation and error reports. The result of the screening is a short list of preferred CGIs and vendors or suppliers.

- (5) Evaluation and Further Grading – The fifth activity consists of a more thorough evaluation of complexity and safety significance with further tailoring of the graded approach to dedication based on the refined vendor and equipment options. The complexity of each design option should be assessed considering the entire system, application, and platform. The evaluation of safety significance in this activity is a continuation of the evaluation from activity three by refining the failure analysis to consider external failure modes of the screened list of preferred items. EPRI TR-107339 stresses the importance of simplicity and avoiding unnecessary complexity to minimize costs, effort, and maintenance requirements.
- (6) Determination of Project-Specific Methods and Activities – This sixth activity defines the specific types of activities required to support evaluation and acceptance of the CGI. Early determination of project-specific methods and activities allows tailoring the project to effectively address project needs including specific requirements for dedication activities. This activity also selects project team members with the required expertise.
- (7) Cost-Benefit Evaluation – This seventh activity assesses the cost/benefit of using commercial-grade digital equipment. EPRI TR-107339 states that a typical cost/benefit analysis is difficult for digital equipment because the benefits are qualitatively known (e.g., reduced number of trips, reduced radiation exposure, reduced operating and maintenance costs) but difficult to quantify. In many cases, a decision is based on added capability necessary to meet regulatory requirements or inability to maintain legacy systems.
- (8) Iteration to Define Project and Options – The eighth activity simply acknowledges that the first seven activities are iterative. The number of products and list of potential vendors are continually refined until a decision can be made to proceed with a narrow, well-defined list of equipment options and suitable vendors.

An ISL observation, based on our analysis of the project definition process, is that the guidance for determining the safety significance of the CGI is general and relies on significant engineering judgment. Although EPRI TR-107339 mentions IEC 61508 and IEC 61226 for classifying systems based on importance to safety which would remove most of the engineering judgment from the activity, EPRI TR-107339 does not recommend the IEC methods. Instead, the primary guidance of EPRI TR-107339 is to define the safety classification by satisfying the requirements of IEEE 603 and Chapter 7 of Standard Review Plan, NUREG 0800, which do not utilize an explicit safety classification technique. NUREG 0800, Chapter 7, does apply a level of grading by singling out the reactor protection system and engineered safety features actuation system for special requirements beyond those applied to other safety systems; however, this graded approach falls well short of a formal safety classification methodology. Based on our analysis, ISL recommends that the NRC consider influencing a change to IEEE 603 or updating NUREG

0800 to implement a safety classification methodology similar to that discussed in the IEC 61508 and IEC 61226 but without the probabilistic risk-based approach.

EPRI TR-107339 also provides supplemental guidance for defining requirements, defining critical characteristics, and verifying critical characteristics using the four methods of EPRI NP-5652 (i.e., testing, conducting surveys and design reviews, source verification, and use of operating history). EPRI TR-107339 states that a large fraction of the problems characterized as software errors are attributed to errors, omissions, inconsistencies, and ambiguities in the requirements specification; therefore, for any digital equipment application, system requirements are very important and should be developed, analyzed, and tracked using a controlled procedure such as that described in EPRI TR-108831. However, EPRI TR-107339 also states that requirements are not perfect and that excessive effort should not be expended refining the requirements document. An ISL observation, based on our analysis of these statements, is that this guidance could be considered contradictory in that requirements are a primary source of software errors but that effort expended on defining the requirements should be limited since the requirements specifications can never be perfect. Another ISL observation is that a more positive interpretation of the guidance could be that a significant but not excessive effort needs to be applied to ensure the requirements specification is as complete as practical while minimizing project risk. The discussion of requirements specification in EPRI TR-107339 should be clarified to indicate which interpretation is more accurate. EPRI TR-107339 also states that the key area of concern with digital systems is the potential for unforeseen or unexpected failure modes and that the requirements specification should address as many of the key areas of risk as practical.

Supplemental guidance for defining critical characteristics consists of identifying the primary differences between dedicating analog and digital equipment and identifying required expertise for defining the critical characteristics. With respect to analog to digital differences, EPRI TR-107339 states that digital equipment is more complex and verification of critical characteristics (e.g., time response) is more difficult. More extensive testing, use of operating history, and examination of the internal data processing architecture may be required to verify critical characteristics. EPRI TR-107339 also states that the dependability of digital equipment is a bigger issue than with analog because the output behavior of digital equipment is more difficult to verify due to the number and complexity of the possible paths an input signal must take through the software processing. Further guidance in EPRI TR-107339 states that all critical characteristics must be verified to commercially dedicate an item; therefore, the list of critical characteristics should be limited to only those characteristics strictly necessary for dedication of the item. With respect to expertise, EPRI TR-107339 states that in addition to personnel with experience in design, procurement, audits, and surveys, digital systems specialists are required to define the critical characteristics and determine the appropriate verification methods.

Supplemental guidance for inspection and testing (i.e., Method 1 of EPRI NP-5652) covers the types of testing necessary for digital equipment and the use of test plans and digital system models. EPRI TR-107339 states that Method 1 testing alone is insufficient to verify the critical characteristics of digital equipment and that a combination of Method 1 testing, Method 2 commercial-grade survey, and Method 4 equipment performance record or operating history is

required. EPRI TR-107339 states that special tests designed to address specific digital issues are required in addition to typical “black box” functional and performance testing. Testing needs to focus on the highest-risk areas or on the most critical parts of the systems. Vendor testing should be reviewed and supplemented by factory acceptance testing, site acceptance testing, post-installation testing, startup testing, and any special tests needed to verify the critical characteristics. EPRI TR-107339 also states that no test program can provide complete assurance for digital equipment and that testing must be coupled with system models. For simple systems, the model may be an analysis or description of the architecture. For complex systems, computer models are used to gain sufficient understanding of the system to be able to interpret the test results.

Supplemental guidance for surveys and design reviews (i.e., Method 2 of EPRI NP-5652) acknowledges that the dependability of software cannot be verified by inspection and testing alone and that the vendor process for developing the software is critical to ensure that the final product meets its requirements under all operating conditions. EPRI TR-107339 states that a survey of the vendor’s software development and QA practices serve to verify critical characteristics related to built-in quality and dependability. However, typical surveys are insufficient to gain the high degree of assurance required for critical software and these surveys must be supplemented by a critical design review and failure analysis, considering abnormal conditions and events and unusual failure conditions that can occur with digital systems. EPRI TR-107339 also states that vendor certification to the ISO 9000 or other international standard is a step in the right direction but not a guarantee that the digital equipment developed and manufactured by a certified vendor is equivalent to an item developed specifically in accordance with the 10 CFR Part 50, Appendix B, QA program. A survey of the vendor’s practices and the certification report are necessary to determine that the vendor met the requirements and how the requirements compare to requirements for commercial-grade dedication.

Supplemental guidance for source verification (i.e., Method 3 of EPRI NP-5652) acknowledges that this method has the least applicability to digital equipment since the built-in software in a commercial device has already been developed and cannot be visually inspected. However, this method may still be used to monitor the installation of the software or firmware into the specific units being supplied or to monitor quality control of software modifications.

Supplemental guidance for use of operating history (i.e., Method 4 of EPRI NP-5652) acknowledges that operating history alone is not sufficient to verify the critical characteristics of digital equipment used in commercial nuclear applications; however, the use of commercial digital equipment should take advantage of the successful and relevant operating experience gained by its wide application in other industries. EPRI TR-107339 states that the review of operating history should focus on areas where the history data is needed to verify critical characteristics and not just the general functionality of the digital equipment. EPRI TR-107339 identifies that the primary sources of operating history data are from the manufacturer or supplier, use within the dedicating utility, other customers who have used the item, and user groups or industry reports. EPRI TR-107339 cautions that successful operating history in other industries may not be meaningful unless the history is relevant to the planned nuclear plant

application. Also, operating history that is not documented cannot be used for credit in a commercial-grade dedication process.

An ISL observation, based on our analysis of EPRI TR-107339, is that the supplemental guidance provided is particularly relevant to commercial-grade software intended for use in a commercial nuclear power application; however, most of the supplemental guidance is not particularly relevant to software tools, especially the failure mode analysis and the evaluation of the tool under abnormal conditions. Software tools are usually executed under ideal conditions and any abnormal event or conditions that occur during testing or use can be eliminated simply by retesting or re-executing the tool. Therefore, the Method 1 verification of critical characteristics using testing may be more effective for software tools than for embedded software. However, Method 1 should still be supplemented, if possible, with vendor surveys and audits of the software tool development process and tool operating history to verify critical characteristics as part of the commercial-grade dedication process. ISL further observes that the graded approach in determining the scope of activities required to commercially dedicate digital equipment should be more prescriptive with less reliance on engineering judgment. Also the classifications of systems should be expanded to multiple categories so that the rigor associated with commercially dedicating safety-related equipment can be more effectively tailored to its importance.

2.9.1.5 *EPRI TR-1025243*

EPRI TR-1025243 provides guidance for the safety classification and acceptance of commercial-grade design and analysis tools used in nuclear safety-related applications via the commercial-grade dedication process. Revision 1 of EPRI TR-1025243 is a minor update to the original report to clarify the intent of using an equivalency evaluation to determine the suitability of proposed replacement software tools that are not identical to the original and to clarify that nonsafety-related design and analysis tools are not required to be procured as basic components or CGIs and dedicated. The analysis of EPRI TR-1025243 applies to both the original report and Revision 1.

EPRI TR-1025243 provides a method for determining the functional safety classification of design and analysis tools that are not resident or installed as part of plant SSCs. EPRI TR-1025243 also provides guidance for using a commercial-grade dedication methodology to accept safety-related, commercial-grade design and analysis tools used in the design and analysis of safety-related plant SSCs. EPRI TR-1025243 extends the commercial-grade dedication process for hardware components described in EPRI NP-5652 to design and analysis software tools used to support the hardware design and development. EPRI TR-1025243 guidance does not apply to all software tools used to support the design and development of safety-related DI&C software. However, with minor modifications to the safety classification and acceptance methodologies, the guidance could be extended to all software design and development tools.

EPRI TR-1025243 identifies three methods of procuring and using design and analysis software tools in safety-related applications. The first method involves the procurement of the software tool as a basic component from a supplier or developer that maintains a nuclear QA program

that meets the requirements of 10 CFR Part 50, Appendix B. In this method, the software tool was either developed or dedicated for use in a safety-related application under the supplier's nuclear QA program. The software tool procured as a basic component using this method is accepted by the purchaser for a defined scope, used within established boundaries, and maintained under configuration control.

The second method involves the procurement of a design or analysis software tool as a CGI from a supplier or developer that does not maintain a QA program that meets the requirements of 10 CFR Part 50, Appendix B. In the second method, the focus is on the output produced by the tool. The commercially procured software tool is accepted for use in accordance with the purchaser's QA program; however, each time the software tool is used, the output of the tool is verified through alternative means such as hand calculations, calculations using comparable proven software tools, or empirical data and information from literature. Commercial design and analysis software tools procured and used in this manner are classified as nonsafety-related because the output from the software tool is verified each time the tool is used and the tool is not relied upon as the sole basis for the design or analysis calculation.

The third method also involves the procurement of a design or analysis software tool as a CGI from a supplier or developer that does not maintain a QA program that meets the requirements of 10 CFR Part 50, Appendix B. However, in the third method, the focus is on the ability of the design and analysis software tool and not the output. The commercially procured design or analysis software tool is dedicated for use as a basic component using one or more of the four methods of dedication described in EPRI NP-5652 prior to use in safety-related design and analysis applications. Commercial design and analysis software tools procured and dedicated in this manner are classified as safety-related because the output of the software tool is relied upon as the sole basis for the design or analysis calculation.

The remainder of the report provides guidance for the commercial-grade dedication process required by the third method of procurement. An ISL observation, based on our analysis of these procurement methods, is that most software development tools of interest to this research project would be procured as CGIs due to a vendor's reluctance to incur the significant cost associated with maintaining a 10 CFR Part 50, Appendix B, QA program and due to the fact that the tools are targeted for non-nuclear, nonsafety-related applications that do not require an equivalent 10 CFR Part 50, Appendix B, QA program.

Commercial-grade dedication involves a technical evaluation followed by an acceptance process to provide a reasonable assurance that the software tool procured meets specified requirements. The technical evaluation for a design or analysis computer program consists of the following activities:

- (1) Identify the scope of use and function of the computer program
- (2) Determine the safety classification of the computer program
- (3) Identify the safety functions of the computer program

- (4) Perform a failure modes and effects analysis to identify characteristics of the computer program
- (5) Identify critical characteristics and acceptance methods
- (6) Specify technical, quality, and documentation requirements
- (7) Document the technical evaluation

EPRI TR-1025243 defines functional safety classifications of computer programs based on the discussion in 10 CFR Part 21. There are only two major classifications: safety-related and nonsafety-related. A subset of nonsafety-related computer programs may be classified as augmented quality. A computer program is safety-related if it affects safety functions necessary to assure: (a) the integrity of the reactor coolant pressure boundary; (b) the capability to shut down the reactor and maintain it in a safe shutdown condition; or (c) the capability to prevent or mitigate the consequences of accidents that could result in potential offsite exposures comparable to those referred to in 10 CFR 50.34(a)(1) or 10 CFR 100.11. Computer programs that are not safety-related are classified as nonsafety-related. An ISL observation, based on our analysis of the safety classification method, is that the simple use of two classifications results in all safety-related computer programs being subject to the same requirements and level of QA even though computer programs have various degrees of potential impact on safety-related components. An expanded classification system that differentiates between software used for development, verification, validation, QA, design, and analysis of safety-related SSCs should be considered by the NRC in any new regulatory guidance so that the rigor of commercial-grade dedication could be tailored commensurate with the importance of the computer program to the safety of the nuclear power plant.

The safety classification is determined on a case-by-case basis by evaluating the actual functions of the computer program and its intended end uses. EPRI TR-1025243 identifies two deterministic options for classifying computer programs. One option considers failure modes and effects. The second option considers the impact of the computer program on SSCs.

The failure modes and effects option for classifying computer programs analyzes and extrapolates the failure of the computer program to determine if it could also result in failure of plant equipment that the computer program is being used to design or analyze. EPRI TR-1025243 states that computer programs that are integral to or impact safety-related SSCs are safety-related. Computer programs can impact SSCs when they facilitate the design of safety-related SSCs, analyze the function of safety-related SSCs, or monitor safety-related SSCs. EPRI TR-1025243 states that computer programs involved in QA aspects of the program (e.g., document control, records management, corrective action tracking, maintenance of training records, emergency response preparedness) are nonsafety-related but augmented quality controls should be considered. EPRI TR-1025243 also states that if the results of computer programs are independently verified each time they are used, the computer program is nonsafety-related.

The impact characterization option for classifying computer programs considers the effect that the software has on the ability of a safety-related SSC to perform its intended safety function. As a means to address the impact of computer programs, EPRI TR-1025243 discusses the methodology of the Nuclear Information Technology Strategic Leadership (NITSL) organization. NITSL-SQA-2005-02, Revision 1, outlines a method for classification of software into one of four categories depending on the software's impact on safety-related SSCs.

- (1) High Impact: Software that has a direct active effect on the ability of a safety-related SSC to perform its intended safety function including software used for the design of SSCs without using alternative methods to verify the results.
- (2) Medium Impact: Software used to assess the ability of SSC to meet its intended safety function and the output of which has been independently verified or software used to monitor operation and control functions of plant SSCs.
- (3) Low Impact: Software used to support activities that have no direct impact on nuclear operations, design, or license commitments but may be used to monitor compliance or optimize performance.
- (4) Other: Software not included in one of the above three classifications.

EPRI TR-1025243 does not recommend implementation of the NITSL methodology but simply mentions the methodology for comparative purposes. NITSL-SQA-2005-02, Revision 1 suggests that all elements of 10 CFR Part 50, Appendix B, be implemented for the high-impact classification which effectively classifies this type of software as safety-related. NITSL-SQA-2005-02, Revision 1 also suggests that the medium- and low-impact classifications use a graded approach and implement only selective criteria from 10 CFR Part 50, Appendix B, effectively classifying this software as nonsafety-related, augmented quality. The fourth category, other, corresponds to nonsafety-related software. An ISL observation, based on our analysis of the NITSL-SQA-2005-02, Revision 1, discussion, is that implementation of the expanded classification system of NITSL-SQA-2005-02, Revision 1, for commercial nuclear power would be a positive step toward recognizing that not all safety-related software tools have an equal impact on safety-related SSC functionality. However, the implementation should be more prescriptive by providing guidance on which "selective criteria" of 10 CFR Part 50, Appendix B, applies to each software classification rather than relying on the engineering judgment of different organizations using the methodology.

Commercial-grade dedication also involves an acceptance process to provide a reasonable assurance that the software tool procured meets specified requirements. EPRI TR-1025243 provides supplemental guidance for each of the four acceptance methods of EPRI NP-5652 specifically for design and analysis computer programs. EPRI TR-1025243 emphasizes that commercial-grade dedication is an acceptance process focused on the verification of critical characteristics defined by the technical evaluation and is not intended to replace other key processes for computer programs such as design, product selection, and establishing suitability

for use. EPRI TR-1025243 guidance for each of the four acceptance methods is discussed in the following paragraphs.

- (1) Method 1 – Special Tests and Inspections. Testing of design and analysis computer programs should ensure that the selected critical characteristics of the computer program are verified. Testing should include run tests installed in its operating environment or system and respective hardware. Acceptance of COTS computer programs typically rely heavily on testing in the installed environment since it is usually not possible to verify critical characteristics by survey or source verification throughout the development life cycle.
- (2) Method 2 – Survey of Commercial-Grade Supplier. A commercial-grade survey of the supplier of design and analysis computer programs may not be a viable method of evaluating commercial quality controls implemented throughout the entire life cycle of the design and analysis computer program being dedicated especially if the computer program has already been developed prior to beginning the dedication process. However, a commercial-grade survey of the supplier's processes can provide an opportunity to develop confidence in the processes that were used to develop the computer program.
- (3) Method 3 – Source Verification. Source verification is typically performed in conjunction with key milestones during the computer program development life cycle. Design and analysis computer programs are primarily COTS items and it is unlikely that source verification is a viable option to verify critical characteristics.
- (4) Method 4 – Supplier and Item Performance History. Performance history determines which of the other three methods of acceptance are appropriate and the rigor to which they are applied. EPRI TR-1025243 does not provide any supplemental guidance for design and analysis computer programs.

An ISL observation, based on our analysis of the supplemental guidance for methods of acceptance of design and analysis computer programs, is that commercial-grade surveys and source verification are not viable options for verifying critical characteristics of COTS computer programs. These methods are not viable because dedication of the computer programs occur well after the programs have been developed and because of the unwillingness of COTS program vendors to allow outside access to proprietary processes and documentation. Therefore, critical characteristics must be verified using extensive testing and performance history of the computer program. The guidance contained in EPRI TR-1025243 for design and analysis software tools can be extrapolated to other software tools; however the limited software safety classifications and lack of specific commercial-grade dedication activities applicable to different types of software tools limits the effectiveness of the guidance.

2.9.2 Summary

Four EPRI documents focus on commercial-grade dedication of hardware used in nuclear safety-related applications while one EPRI document provides guidance for commercial-grade

dedication of software used in nuclear safety-related applications. The commercial-grade dedication process discussed in these EPRI documents should form a sound technical basis for developing new software tool regulatory guidance when supplemented with guidance from other industries.

The five EPRI reports analyzed discuss utilization of CGIs and commercial-grade dedication of equipment for nuclear safety-related applications. Although most of the reports are focused on hardware and software, the commercial-grade dedication process described in these reports can be applied to software tools used in the development of safety system software. EPRI NP-5652 contains a generic process for the acceptance of commercial-grade hardware components and parts for use in a nuclear safety-related application. The most important part of the generic process is accepting the item. There are four acceptance methods: special tests and inspections, commercial-grade survey of the supplier, source verification, and acceptable supplier or item performance record. The four acceptance methods were conditionally endorsed by the NRC in GL 89-02, with restrictions on the use of supplier surveys and item performance history as a sole method of acceptance. NRC GL 89-02 states that supplier surveys should not be employed as the sole basis for accepting items from suppliers with undocumented commercial quality control programs or with programs that do not effectively implement their own necessary controls. The generic process and acceptance methods were written for applicability to hardware components and parts; however, the process could be applied to software, as discussed in EPRI TR-1025243, and by extension, software tools.

EPRI TR-102260 contains supplemental guidance for dedication of CGIs since EPRI NP-5652 was issued. A fundamental concept of commercial-grade dedication discussed in EPRI TR-102260 is in the application of a graded approach that is consistent with 10 CFR Part 50, Appendix B, Criterion II which requires that a QA program control activities affecting the quality of the identified SSCs, to an extent consistent with their importance to safety. An ISL observation, based on our analysis of this fundamental concept, is that the graded approach should be extended to the functional safety classification of CGIs and expanded beyond the guidance in EPRI NP-5652 which classifies a component as either safety-related or nonsafety-related. A graded approach using four or five integrity levels would still be consistent with EPRI NP-5652 in that all safety-related components would need to be dedicated; however, the graded approach would subject the most important safety-related items to a more rigorous dedication process than less important safety-related items. EPRI TR-102260 does not contain any specific discussion of digital equipment, software, or software tools. However, the commercial-grade dedication process can be applied to software and software tools with minor clarifications of the process.

EPRI TR-106439 contains guidance on the evaluation and acceptance of commercial-grade digital equipment as part of the commercial-grade dedication process for nuclear safety applications. EPRI TR-106439 relies on the existing commercial-grade dedication process described in EPRI NP-5652 and EPRI TR-102260 with supplemental guidance to address digital-specific issues. Since EPRI NP-5652 applies primarily to hardware, EPRI TR-106439 states that new critical characteristics and additional verification activities are needed when the methods of EPRI NP-5652 are applied to digital devices. EPRI TR-106439 also states that no

one method from EPRI NP-5652 will suffice by itself and that for many digital devices, a combination of methods is needed. EPRI TR-106439 discusses digital equipment critical characteristics; however, most of the critical characteristics mentioned do not apply to software tools. Commercial dedication of software tools is not directly addressed in EPRI TR-106439. Many of the steps described in the dedication process would be difficult to accomplish if applied to software tools. An ISL observation, based on our analysis of EPRI TR-106439, is that the burden of commercial dedication of software tools will fall to the vendors developing the final software, who will be using the tools, or to the licensee.

EPRI TR-107339 contains supplemental guidance for evaluating commercial digital equipment for nuclear safety applications and other applications requiring high reliability through tailoring of the guidance in EPRI TR-106439 consistent with the importance of the digital equipment to safety and economics. The primary focus of EPRI TR-107339 is on the differences in the technical evaluation and acceptance process required for digital, software-based equipment and systems. EPRI TR-107339 provides guidance for each of the phases of the CGI dedication process including project definition, defining the detailed requirements, defining the critical characteristics for acceptance, formulation of an acceptance strategy, and verification of critical characteristics. The supplemental guidance of EPRI TR-107339 is particularly relevant to commercial-grade software but not particularly relevant to software tools. An ISL observation, based on our analysis of EPRI TR-107339, is that the commercial-grade dedication of digital equipment should place less reliance on engineering judgment and the classifications of systems should be expanded beyond safety-related and nonsafety-related so that the rigor of dedication can be better tailored to the importance of the commercial item.

EPRI TR-1025243 provides guidance for the safety classification of design and analysis tools that are not resident or installed as part of plant SSCs. EPRI TR-1025243 also provides guidance for using a commercial-grade dedication process to accept safety-related, commercial-grade design and analysis tools used in the design and analysis of safety-related plant SSCs. EPRI TR-1025243 extends the commercial-grade dedication process described in EPRI NP-5652 from hardware components to design and analysis software tools used to support the hardware design and development. EPRI TR-1025243 guidance does not apply to all software tools used to support the design and development of safety-related DI&C software.

EPRI TR-1025243 defines two major classifications of computer programs: safety-related and nonsafety-related based on 10 CFR Part 21. EPRI TR-1025243 also defines a subset of nonsafety-related computer programs as augmented quality. EPRI TR-1025243 clarifies that computer programs involved in QA aspects of the program (e.g., document control, records management, corrective action tracking, maintenance of training records, emergency response preparedness) are nonsafety-related; however, augmented quality controls should be considered. EPRI TR-1025243 clarifies that computer programs that are independently verified each time they are used are nonsafety-related. EPRI TR-1025243 also clarifies that nonsafety-related design and analysis tools are not required to be procured as basic components or CGIs and dedicated. EPRI TR-1025243 defines two methods of determining the classification of computer programs. The first method is a failure modes and effects analysis. The second method is to determine the impact of the computer program on SSCs. The impact

characterization does not formally endorse but mentions a NITSL-SQA-2005-02 four category method of classifying software: high impact, medium impact, low impact, and other. An ISL observation, based on our analysis of the NITSL-SQA-2005-02 four category method, is that high impact corresponds to safety-related software, medium and low impact correspond to nonsafety-related, augmented quality, and other corresponds to nonsafety-related. The guidance contained in EPRI TR-1025243 for design and analysis software tools for designing safety-related hardware could be extrapolated to software tools used to design and analyze DI&C software; however the limited software safety classifications and lack of specific commercial-grade dedication activities applicable to different types of software tools limits the effectiveness of the guidance.

EPRI technical reports provide a comprehensive hardware and software commercial-grade dedication process that can be adapted for software tools. Based on the analysis of EPRI documents, the following essential elements of the documents relative to software tools have been determined.

- (1) EPRI NP-5652 contains a generic process for the acceptance of CGIs for use in a nuclear safety-related application.
- (2) Acceptance of a CGI is based on one or more alternative methods including special tests and inspection, commercial grade survey of supplier, source verification, and acceptable supplier/item performance record.
- (3) The use of engineering judgment and the concept of reasonable assurance in EPRI NP-5652 and EPRI TR-102260 are accepted techniques in the application of the graded approach to commercial-grade dedication consistent with the importance of the item to safety.
- (4) The safety classification of a component is determined from an analysis of its safety function and is categorized as either safety-related or nonsafety-related.
- (5) EPRI TR-106439 emphasizes identification of digital equipment critical characteristics with subsequent verification through testing, analysis, vendor assessments, and review of DI&C operating experience.
- (6) EPRI TR-106439 states that engineering judgment must be based on quantifiable critical characteristics and adequate evidence.
- (7) Documentation of judgment should be sufficient to allow a comparably qualified individual to come to the same conclusion.
- (8) EPRI TR-107339 does not recommend the IEC 61226 method of classifying systems based on importance to safety. EPRI TR-107339 defines the safety classification of systems important to safety by satisfying the requirements of IEEE 603 and Chapter 7 of Standard Review Plan, NUREG 0800.

-
- (9) EPRI TR-107339 states that digital equipment is more complex and verification of critical characteristics (e.g., time response) is more difficult. More extensive testing, use of operating history, and examination of the internal data processing architecture may be required to verify critical characteristics.
 - (10) EPRI TR-107339 states that all critical characteristics must be verified to commercially dedicate an item; therefore, the list of critical characteristics should be limited to only those characteristics strictly necessary for dedication of the item.
 - (11) EPRI TR-107339 states that vendor certification to the ISO 9000 or other international standard is not a guarantee that the digital equipment developed and manufactured by a certified vendor is equivalent to an item developed specifically in accordance with the 10 CFR Part 50, Appendix B, QA program.
 - (12) EPRI TR-1025243 provides guidance for using a commercial-grade dedication methodology to accept safety-related, commercial-grade design and analysis tools used in the design and analysis of safety-related plant SSCs.
 - (13) EPRI TR-1025243 states that if the results of a computer program are independently verified each time the program is used, the computer program is nonsafety-related because the tool is not relied upon as the sole basis for the design or analysis calculation.
 - (14) EPRI TR-1025243 defines functional safety classifications of computer programs as either safety-related or nonsafety-related; however, a subset of nonsafety-related computer programs may be classified as augmented quality.

Based on the analysis of EPRI documents, ISL has made the following important observations and recommendations.

- (1) The EPRI NP-5652 safety function process can be adapted to software tools by determining the safety function of a software tool based on the classification of the computer-based component being developed with the tool.
- (2) Consistent with current NRC guidance, the NRC should consider that regulatory guidance should not require nonsafety-related software tools to be dedicated consistent with their hardware counterparts.
- (3) The EPRI NP-5652 commercial grade confirmation process can be adapted to software tools by confirming that the tool has not been developed in accordance with a formal and recognized software life cycle development process, is used in non-nuclear computer-based applications, and is considered off-the-shelf and purchased based on a catalog description.

-
- (4) The EPRI NP-5652 critical characteristic process can be applied to software tools by determining the tool's version number, operational environment, functional requirements, and the output produced by the tool.
 - (5) Acceptance Method 1, Special Tests and Inspections, can be applied to software tools by physical inspections of software tool executables including a verification of the tool version, installation package, intended operational environment, development environment, and testing in the installed environment.
 - (6) Acceptance Method 2, Commercial-Grade Survey of Supplier, can be applied to software tools by surveying the software tool developer's life cycle development process including the supporting QA, V&V, and configuration control processes.
 - (7) Acceptance Method 3, Source Verification, can be applied to software tools by verification of the software tool requirements specification, design requirements, and code requirements and direct observation of QA audits and reviews except for COTS tools where it is unlikely that source verification is a viable option.
 - (8) Acceptance Method 4, Acceptable Supplier/Item Performance Record, is the most reliable method to accept COTS software tools through an examination of defect reports and documentation of successful use of the tool in other applications. However, the use of performance records to accept complex software tools should be done with extreme caution.
 - (9) Engineering judgment can be relied upon to establish reasonable assurance that the item received meets specified requirements and for selecting the best EPRI NP-5652 method of verifying the critical characteristics; however, engineering judgment should not be relied upon to eliminate commercial-grade dedication processes (e.g., failure analysis).
 - (10) Vendors that produce COTS software tools would likely not be interested in the time and expense of the commercial dedication process given the relatively small market. Therefore, the burden of commercial-grade dedication of software tools will fall to the vendors developing the final software, who will be using the tools, or to the licensee.
 - (11) IEEE 603 or NUREG 0800 should implement a safety classification methodology similar to that discussed in the IEC 61508 and IEC 61226 but without the probabilistic risk-based approach.
 - (12) EPRI TR-107339 guidance is not particularly relevant to software tools, especially the failure mode analysis and the evaluation of the tool under abnormal conditions. Software tools are usually executed under ideal

- conditions and any abnormal event or conditions that occur during testing or use can be eliminated simply by retesting or re-executing the tool.
- (13) The EPRI TR-107339 graded approach in determining the scope of activities required to commercially dedicate digital equipment should be more prescriptive with less reliance on engineering judgment.
 - (14) EPRI TR-1025243 guidance does not apply to all software tools used to support the design and development of safety-related DI&C software. However, with minor modifications to the safety classification and acceptance methodologies, the guidance could be extended to all software design and development tools.
 - (15) The EPRI TR-1025243 use of two software classifications (i.e., safety-related and nonsafety-related) results in all safety-related computer programs being subject to the same requirements and level of QA even though computer programs have various degrees of potential impact on safety-related components.
 - (16) Implementation of the expanded classification system of NITSL-SQA-2005-02, Revision 1, for commercial nuclear power would be a positive step toward recognizing that not all safety-related software tools have an equal impact on safety-related SSC functionality.
 - (17) The NRC should consider in any new regulatory guidance, an expanded classification system that differentiates between software used for development, verification, validation, QA, design, and analysis of safety-related SSCs.
 - (18) Commercial-grade surveys and source verification are not viable options for verifying critical characteristics of COTS computer programs because dedication of the computer programs occur well after the programs have been developed and because of the unwillingness of COTS program vendors to allow outside access to proprietary processes and documentation.

2.10 National Institute of Standards and Technology Analysis

The National Institute of Standards and Technology (NIST) was established in 1988 to strengthen U.S. innovation and industrial competitiveness by advancing science and engineering and improving public health, safety, and the environment. New research results and NIST technical expertise are communicated by publishing professional journals and technical reports. NIST's Computer Systems Laboratory (CSL) develops standards and guidelines, provides technical assistance, and conducts research for computers and related telecommunications systems to achieve more effective utilization of IT resources. CSL assists agencies in developing security plans and in improving computer security awareness training.

2.10.1 Document Analysis

CSL research and guidance is reported to agencies and organizations in industry, government, and academia through the special publication (SP) 500 series of reports and interagency/internal reports. The most recent special publications issued by CSL discuss advanced computer security and internet congestion control mechanisms. Some earlier special publications and interagency reports discuss verification, validation, and QA of computer software. The only NIST report that provides guidance for software tools is NIST SP 500-234.

2.10.1.1 NIST SP 500-234

NIST SP 500-234 addresses software V&V of computing systems employed in the health care environment in efforts to increase reliability of care and reduce costs. Software V&V is conducted throughout the planning, development, and maintenance of software systems and may assist in assuring appropriate reuse of software. The guidance in NIST SP 500-234 addresses V&V issues related to the recognition that different health care systems may execute in real time, rely on existing software, contain many units contributing to system complexity, and incorporate knowledge-based systems (e.g., diagnostic systems). The software V&V process is tightly integrated with the software development process. For each activity in software development, a corresponding software V&V activity verifies or validates the products of those activities. Knowledge-based systems have been developed to make use of artificial intelligence (AI) techniques to understand the complex medical and patient data used for diagnosis. NIST SP 500-234 provides an overview of the issues in using knowledge-based systems techniques on systems requiring high reliability and provides guidelines on some of the techniques for V&V of knowledge-based systems which employ expert systems.

NIST SP 500-234 discusses the selection of personnel and recommends the use of a separate group for V&V activities, called independent V&V (IV&V). IV&V ensures the objectivity of staff personnel performing software V&V tasks and consists of technical independence, managerial independence, and financial independence. Technical independence requires performance of software V&V by a team that is separate from the development group and have some knowledge about the system design or related experience and engineering background to understand the system. With respect to software tools, NIST SP 500-234 recommends that the IV&V team execute qualification tests on tools (e.g., compilers, assemblers, and utilities) shared with the development environment to ensure that the common tools do not mask errors in the software being analyzed and tested. Whenever possible, the IV&V team should use or develop its own set of test and analysis tools. Other than the high-level guidance of developing independent software tools or independently qualifying software tools common with the development environment, NIST SP 500-234 does not categorize tools or use safety as a measure to determine tool qualification requirements nor does NIST SP 500-234 provide guidance for developing independent software tools.

NIST SP 500-234 discusses management of software V&V. One important task of software V&V management is the preparation of a software V&V plan. NIST SP 500-234 recommends that the plan should include details for acquiring tools and for training personnel. NIST SP 500-

234 is not specific about the extent or subject of training; however, the implication is that personnel should be trained in software V&V activities and in the use of tools.

NIST SP 500-234 also discusses the use of artificial intelligence techniques to manage the knowledge base of the medical domain. The term knowledge-based system refers to systems that use complex data or knowledge structures using artificial intelligence techniques. The discussion of knowledge-based systems development addresses the need and use of expert system shells and tools to support rapid encoding of knowledge; however, guidance, verification, validation, and qualification of these expert tools are not discussed.

2.10.2 Summary

NIST SP 500-234 contains only a few high-level software tool requirements that deal with V&V and training and would be of little benefit as a technical basis for developing new regulatory guidance for the commercial nuclear power industry.

NIST guidance for software tools addresses IV&V, planning, and training. The guidance recommends that IV&V personnel develop their own tools or as a minimum, independently qualify V&V tools that are common with the development process. NIST guidance further states that the software V&V plan should discuss tool acquisition and that personnel should be fully trained in the use of tools.

Based on the analysis of the NIST special publication, the following essential elements of the document relative to software tools have been determined.

- (1) An IV&V team should execute qualification tests on tools (e.g., compilers, assemblers, and utilities) shared with the development environment to ensure that the common tools do not mask errors in the software being analyzed and tested.
- (2) Whenever possible, the IV&V team should use or develop its own set of test and analysis tools.
- (3) The software V&V plan should include details for acquiring tools and for training personnel.

Based on the analysis of the NIST special publication, ISL has made the following important observations and recommendations.

- (1) NIST SP 500-234 does not categorize tools or use safety as a measure to determine tool qualification requirements.
- (2) NIST SP 500-234 does not provide guidance for developing independent software tools.
- (3) NIST SP 500-234 is not specific about the extent or subject of training; however, the implication is that personnel should be trained in software V&V activities and in the use of tools.

2.11 Atomic Energy of Canada Limited Analysis

The Atomic Energy of Canada Limited (AECL) is Canada's premier science and technology organization that serves an important public policy role in nuclear matters, providing advice, council, and service as an agent of government. AECL is a world leader in developing peaceful and innovative applications from nuclear technology through its expertise in physics, metallurgy, chemistry, biology, and engineering. One of AECL's core programs is science and technology which includes support for the Canadian-invented Deuterium-Uranium (CANDU®) pressurized heavy water nuclear reactor. AECL works with the CANDU® owners group to provide programs for cooperation, assistance, and exchange of information for support, development, operation, maintenance, and economics of CANDU® -based reactor technology. AECL also supports technology development for the CANDU® industry to implement new technologies that will enhance the safety and economic performance of CANDU® reactors. The AECL maintains a library of published information including books, journals, standards, and technical reports developed through cooperation with Canadian utilities.

2.11.1 Document Analysis

A few AECL standards discuss requirements for software engineering, software development, and software safety classifications. This section discusses an analysis of one of those standards, AECL CE-1001-STD, that specifies requirements for safety-critical software engineering in CANDU® nuclear generating stations. The analysis discussion focuses on the use, selection, verification, validation, and qualification of software tools used to development safety-critical software.

2.11.1.1 AECL CE-1001-STD

AECL CE-1001-STD is a Canadian standard for software engineering of safety critical software jointly prepared by the AECL and Ontario Power Generation, Inc. This standard specifies requirements for the software engineering of safety critical software (i.e., category I software) used in one of the four special safety systems (i.e., two shutdown systems, emergency coolant injection system, and containment system) in CANDU® nuclear generating stations. Requirements for the software engineering of category II and III software are contained in AECL CE-0101-STD. Category I, II, and III software are distinguished by criticality of application, complexity of system, and source of supply. Category I software requires systematic (mathematical) verification, hazards analysis, and reliability qualification and imposes more rigorous requirements on other processes. AECL CE-1001-STD does not apply to systems programmed in function-block diagram languages although the standard can be tailored for that application. AECL CE-1001-STD does not impose requirements on qualification of pre-developed software or to software not originally developed to this standard. The purpose of AECL CE-1001-STD is to define a minimum set of software engineering processes to be followed in creating and revising the safety critical software; define the minimum set of outputs to be produced by the processes, and define the requirements for the content of the outputs.

Section 2 of AECL CE-1001-STD describes a minimum set of software engineering processes that must be part of the software engineering effort for safety critical software. This standard

uses a simplified set of software engineering processes that follow the processes described in AECL CE-0100-STD. Section 2 of AECL CE-1001-STD describes the software engineering process using a simple sequential model consisting of development, verification, validation, and support processes. For each process defined, AECL CE-1001-STD identifies the outputs that must be produced and the requirements that those outputs must satisfy. AECL CE-1001-STD acknowledges that feedback and iterations on the sequential model are expected. Section 2 of AECL CE-1001-STD does not discuss software tool use, guidance, requirements, verification, validation, qualification, review, or approval practices associated with the overall software engineering processes.

Section 3 of AECL CE-1001-STD describes the objectives, inputs, and outputs of the software development process consisting of the requirements definition process, design process, and code implementation process. The requirements definition process analyzes and documents the requirements for the software and identifies and documents any implementation constraints on the software. The design process produces a software architecture and detailed design description that satisfies the requirements from the requirements definition process and identifies self-checks that enhance the robustness of the design to hardware failures or other system level hazards. The code implementation process translates the software design description into source code, translates the source code into executable code, integrates and debugs the executable code, and creates all required databases. Section 3 of AECL CE-1001-STD does not discuss software tool use, guidance, requirements, V&V, qualification, review, or approval practices associated with the software engineering development process.

An ISL observation, based on our analysis of Section 3 of AECL CE-1001-STD, is that software tool requirements should be specified in this section since a significant number of software tools are designed for the software development process and other sections of the standard contain high-level software tool requirements. AECL CE-1001-STD should acknowledge that tools are a major part of the software development process and should require, as a minimum, tool planning, tool identification, tool documentation, and tool qualification as part of the software development process. Also, AECL CE-1001-STD should provide guidance on the selection of tools as an integrated set that will function throughout the life cycle of the software product.

Section 4 of AECL CE-1001-STD contains requirements for the software V&V processes which include the review processes, the systematic verification process, testing, hazards analysis, and reliability qualification. Reliability qualification is a testing methodology using randomized input values. A software tool requirement associated with reliability qualification states that reliability qualification procedures should identify all tools required to produce the input and expected result data for the test cases. Section 4 of AECL CE-1001-STD also discusses common V&V output requirements including review reports, systematic verification reports, test procedures, and test reports. Software tool requirements associated with common V&V output requirements state that all test procedures should identify all tools required to perform the test; all reports should summarize the activities performed and the methods and tools used; and all reports should identify the actual equipment, tools, and support software used. An ISL observation, based on our analysis of these requirements, is that the last two requirements for test reports

are partially redundant and that all of the requirements are simply high-level requirements to adequately identify and document tools used in the software V&V processes.

Section 5 of AECL CE-1001-STD contains requirements for software development support processes including planning, CM, and training. One software tool requirement states that the planning process should establish estimates, schedules, budgets, and resource requirements for developing support tools. Section 5 of AECL CE-1001-STD contains additional software tool requirements associated with planning and documentation in the software development plan. The software development plan is a comprehensive plan for the management of the software engineering process consisting of a project plan, documentation plan, CM plan, and training plan. Software tool requirements associated with each of those plans can be summarized as follows:

- (1) The project plan shall specify tools to be used during development, verification, and support; identify personnel required to maintain and manage the tools; and identify the qualification requirements for software tools for the target system.
- (2) The development plan shall identify tools used to produce, manage, and issue documentation.
- (3) The CM plan shall uniquely identify all categories of configuration items that form a developmental baseline for SCM including support tools (compilers, linkers, operating systems).
- (4) The training plan shall identify requirements for training courses and materials with respect to the use of software tools and methodologies and identify training facilities, tools, responsibility for training, and the contents of the training records.

Section 5 of AECL CE-1001-STD also contains requirements for process, documentation, CM, and training standards and procedures. AECL CE-1001-STD contains two software tool requirements for standards and procedures, both associated with process standard and procedures. One software tool requirement states that the process standards and procedures shall identify procedures for using tools in the development, verification, and support processes. The second software tool requirement states that the process standards and procedures shall identify or reference pertinent documents that identify all verification tools, their proper use, and current versions. An ISL observation, based on our analysis of these requirements, is that all of the requirements are simply high-level requirements to adequately identify and document tools used in the software development process procedures.

2.11.2 Summary

AECL books, journals, standards, and technical reports contain only a few high-level software tool requirements and would be of little benefit as a technical basis for developing new regulatory guidance for the commercial nuclear power industry.

AECL CE-1001-STD specifies requirements for the software engineering of safety critical (i.e., category I) software used in one of the four special safety systems (i.e., two shutdown systems, emergency coolant injection system, and containment system) in CANDU® nuclear generating stations. AECL CE-1001-STD focuses on software development life cycle requirements and contains few software tool requirements. Software tool requirements are limited to high-level requirements to identify necessary resources for tool development, identify and document tools in plans and procedures, identify personnel required to maintain and manage the tools; and identify the qualification requirements for software tools. AECL CE-1001-STD does not distinguish between different types of tools, does not use tool categories or classifications, and does not contain any specific software tool qualification requirements tailored to the type of tool.

Based on the analysis of the AECL standard, the following essential elements of the document relative to software tools have been determined.

- (1) AECL CE-1001-STD specifies requirements for the software engineering of safety critical software (i.e., category I software) used in one of the four special safety systems (i.e., two shutdown systems, emergency coolant injection system, and containment system) in CANDU® nuclear generating stations.
- (2) As part of the software V&V processes, reliability qualification procedures should identify all tools required to produce the input and expected result data for the test cases.
- (3) All test procedures should identify all tools required to perform the test.
- (4) All test reports should summarize the activities performed and the methods and tools used.
- (5) All test reports should identify the actual equipment, tools, and support software used.
- (6) The planning process should establish estimates, schedules, budgets, and resource requirements for developing support tools.
- (7) The project plan shall specify tools to be used during development, verification, and support; identify personnel required to maintain and manage the tools; and identify the qualification requirements for software tools for the target system.
- (8) The development plan shall identify tools used to produce, manage, and issue documentation.
- (9) The CM plan shall uniquely identify all categories of configuration items that form a developmental baseline for SCM including support tools (compilers, linkers, operating systems).
- (10) The training plan shall identify requirements for training courses and materials with respect to the use of software tools and methodologies and identify

training facilities, tools, responsibility for training, and the contents of the training records.

- (11) The process standards and procedures shall identify procedures for using tools in the development, verification, and support processes.
- (12) The process standards and procedures shall identify or reference pertinent documents that identify all verification tools, their proper use, and current versions.

Based on the analysis of the AECL standard, ISL has made the following important observations and recommendations.

- (1) The standard only contains high-level requirements to adequately identify and document tools used in the software development processes.
- (2) Detailed software tool requirements should be added to the standard since a significant number of software tools are designed for the software development process and other sections of the standard contain high-level software tool requirements.
- (3) The standard should acknowledge that tools are a major part of the software development process and should require, as a minimum, tool planning, tool identification, tool documentation, and tool qualification as part of the software development process.
- (4) The standard should provide guidance on the selection of tools as an integrated set that will function throughout the life cycle of the software product.

3 Industry/Organization Comparisons

An ISL observation, based on an analysis of software tool requirements and guidance in various industries and organizations reviewed as part of this project, is that there are significant differences in the way software tools are treated; however, there are also many common requirements throughout these industries and organizations. The common requirements include adequate planning for tool use, configuration management (CM) of tools, verification and validation (V&V) of tools, and adequate documentation of tool use. The differences in the treatment of software tools in the various industries and organizations are in the details associated with establishing the suitability of software tools for developing safety, safety-related, or safety critical systems as the term is used in the various industries. Differences exist regarding which tools are required to be verified and validated, the required rigor of V&V, the use of a software and software tool classification scheme, the treatment of commercial off-the-shelf (COTS) software tools including the required participation of the software tool vendor, the required software tool development process, and in basic terminology. The terms verification, validation, and qualification are used throughout the various industries and organizations and, in most cases, the terms mean the same thing. However, specific industries and organizations use the terms certification, basic component, dedication, safety, and safety critical and these have specific meanings in those industries and organizations.

This section compares requirements, guidance, and review and approval practices of all the different industries and organizations based on the document analysis discussed in Section 2. This section discusses the best practices of each industry or organization and areas that are either not addressed or could be improved in each industry or organization.

3.1 Aerospace Industry

NASA software engineering requirements are primarily concerned with and apply to software that resides within a system or subsystem and which will ultimately be executed on the vehicle, space asset, or ground as part of the mission. The requirements are not written specifically for software tools and the applicability of the requirements to software tools requires a liberal interpretation of words that clarify and expand the requirements. Most of the software tool discussions simply identify and encourage the use of certain types of tools in different software development life cycle processes.

Software is designated as safety-critical or nonsafety-critical based on a software safety litmus test and assigned a classification from Class A to Class H depending on functionality. The NASA safety-critical and nonsafety-critical software designation is consistent with the safety-related and nonsafety-related software classifications used in 10 CFR Part 21. The NASA Class A through H classification of software is different from other industries because it covers all types of software and software tools within the same classification scheme and is based on software functionality rather than the effect of software errors on the safety of the system. Both the software safety litmus test and the software classification system are deterministic processes without regard for system hazards or risk. This deterministic process is similar to that used in the civil aviation industry. Other industries use a probabilistic risk-based process to classify software and software tools.

NASA defines a very comprehensive set of requirements for the development, documentation, and operation of models and simulators but does not address the vast majority of software development computer-aided software engineering (CASE) tools currently in use. Similar to all other industries, NASA requires that software tools be placed under CM and tool use documented in the various plans associated with the software development processes. The NASA standards, guidebooks, and handbooks identify and recommend tools for many of the software engineering processes and require that safety-critical tools be verified, validated, and accredited but lack details on how to verify, validate, or accredit the tool. The aerospace industry has not published documents similar to RTCA DO-178C and RTCA DO-330 that define the qualification requirements for software tools and provide detailed software tool development life cycle processes and activities for developing and qualifying the tool based on the classification level of the tool and the type of tool.

NASA simplifies the software review and approval process through the use of a requirements compliance matrix that lists the applicable software development requirements, the party responsible for satisfying the requirement, and a method by which the reviewer can document full, partial, or non-compliance with the requirement. Although the matrix does not directly apply to software tools, the concept of a requirements compliance matrix is one of the best NASA software development practices and could be extended to software tools with minimum effort.

3.2 Civil Aviation Industry

Both the FAA and RTCA publish documents that address various aspects of verification, validation, certification, qualification, and use of software and software tools. Consistent with other industries, the FAA requires that all software tools be identified, validated, and addressed within the software development activities and documentation. Software is classified (i.e., Levels A through E) based on how an error affects the software and system containing the software. The software classification defines the rigor necessary to demonstrate compliance with software development requirements. The civil aviation industry software classification system is different from the aerospace industry because the classifications are based on effects of anomalous behavior of the software rather than software functionality. Unlike the aerospace industry which assigns software tools a classification separate from embedded system software, the civil aviation industry classifies software tools to the same level as the software developed with the tool. The civil aviation industry software classification system is also similar to the method used by the international commercial nuclear power industry practice described in IEC 61508 and associated standards, the automotive industry practice described in ISO 26262, the high-speed railway industry practice described in BS EN 50128, and the method used in IEEE 1012-2004. However, the civil aviation industry practice is based on a deterministic assessment of software errors and their impact on aircraft, crew, and system safety compared to the probabilistic risk-based methods of IEC 61508, ISO 26262, and BS EN 50128. These software classification methods are different from the guidance in RG 1.152 which assigns the highest safety integrity level (SIL) to software executed in a commercial nuclear power plant.

Unlike the aerospace industry which does not provide guidance for the qualification of software tools, the civil aviation industry, specifically the RTCA, defines when software tools require

qualification and the level of rigor necessary for the qualification based on the software tool classification level and the type of tool. The civil aviation industry practice of defining the requirements for tool qualification is similar to the international commercial nuclear power industry practice described in IEC 61226 and IEC 60880, the automotive industry practice described in ISO 26262, and the high-speed railway industry practice described in BS EN 50128. However, these other industries do not provide a comprehensive tool qualification process analogous to the process provided in RTCA DO-330.

Tool qualification requirements are determined in RTCA DO-178C by first determining the type of tool. RTCA DO-178C uses three criteria to determine the impact of the tool on software life cycle processes and essentially differentiates between software development tools and various types of software verification tools. The three criteria in RTCA DO-178C are similar to tool classes of T1, T2, and T3 in IEC 61508 and BS EN 50128. The three criteria for determining the impact of tools in RTCA DO-178C are also similar to the tool impact class and tool defect class used in ISO 26262. The tool criteria along with the software classification level are used in RTCA DO-178C to select a tool qualification level (TQL) to determine the necessary rigor of qualification. The TQL in RTCA DO-178C is similar to the tool confidence level (TCL) used in ISO 26262 to select an appropriate combination of methods to qualify the software tool. The determination of the software level and the type of tool are both deterministic processes that assess how potential failures could impact the aircraft and crew. Risk and probability are not used in the FAA tool qualification process.

RTCA DO-330 contains a complete tool qualification and development process based on the RTCA DO-178C software development process but tailored for software tools. The primary method of tool qualification is development, verification, and validation in accordance with a high-quality and well-organized software tool development life cycle process. This tool qualification method is similar to the optional software tool validation method specified in ISO 26262 and the optional tool testing and validation method specified in IEC 60880. RTCA DO-330 also provides alternative tool qualification methods of using service history, exhaustive input testing, formal methods, and use of dissimilar tools. These alternative methods are analogous to commercial-grade item (CGI) dedication when there is not much information available from the vendor beyond the published product description and a user's manual, and commercial-grade surveys (EPRI Method 2) or source verifications (EPRI Method 3) are impractical leaving only special tests and inspections (EPRI Method 1) and use of supplier and product performance history (EPRI Method 4). The use of service history is analogous to EPRI Method 4 in nuclear commercial-grade dedication. Unlike most other industries that superficially address COTS software tools, RTCA DO-330 provides a comprehensive qualification process for COTS tools that divides qualification responsibilities between the COTS developer and tool user.

The tool CM process described in RTCA DO-330 is similar to the processes described in IEEE 1042-1987 and IEEE 828-2012. It also reinforces the need for CM of software tools as described in all versions of IEEE 7-4.3.2. The tool quality assurance (QA) process described in RTCA DO-330 has some of the requirements of 10 CFR Part 50, Appendix B, including the requirement that all tool QA objectives be satisfied with independence. In general, the process

described is similar to that described in IEEE 730-2002 and IEEE 730-2014 without the depth of coverage required in the documentation. The tool qualification liaison process described in RTCA DO-330 contains similar activities and tasks as Interim Staff Guidance (ISG) 06 to facilitate the review and approval of software tools. RTCA DO-330 does not contain a requirements compliance checklist to facilitate the review and certification process comparable to the aerospace industry. The tool qualification data described in RTCA DO-330 is the minimum information necessary to describe and document the tool qualification process similar to the data required in the various plans described in Branch Technical Position (BTP) 7-14.

3.3 Nuclear Regulatory Commission

The NRC generally classifies nuclear facility structures, systems, and components (SSCs) as either important to safety or not important to safety. As defined in 10 CFR 50.49(b), important-to-safety SSCs comprise (b)(1) safety-related SSCs (See definition in the glossary), (b)(2) nonsafety-related SSCs, whose failure could impact one or more safety functions, and (b)(3) certain post-accident monitoring (PAM) equipment (RG 1.97). However, similar to the aerospace industry which determines the safety implications of software using a software safety litmus test, the NRC uses a two-level safety classification system of safety-related and nonsafety-related, which is prescribed by 10 CFR 50.55a(h) in requiring compliance with IEEE 603-1991 and endorsed in RG 1.152 through its endorsement of IEEE 7-4.3.2-2003. However, unlike most industries that use multiple SILs to determine the minimum software verification, validation, or QA activities, the NRC does not endorse that type of scheme for safety-related software and instead requires that safety-related software used in a commercial nuclear power plant be assigned the highest SIL. This approach appears to neglect software that would fall under 10 CFR 50.49(b)(2), but the apparent discrepancy is rendered moot by the NRC requirement that nonsafety-related software executed on the same computer as safety-related software also be treated as safety-related and assigned the highest SIL. Thus, important-to-safety software is treated as safety-related software, presumably because of the potential for the failure or malfunction of nonsafety-related software (i.e., no safety functions) to impact the performance of safety-related equipment in which it resides. With regard to PAM equipment software, although 10 CFR 50.49(b)(3) requires certain PAM equipment to be environmentally qualified, the regulations do not address the safety classification of PAM equipment software as such. An ISL observation, is that it is reasonable that PAM equipment software, being a type of nonsafety-related software that normally does not reside in or control safety systems, could be treated as nonsafety-related. Nevertheless, the treatment of all software important to safety as SIL 4 is inconsistent with 10 CFR Part 50, Appendix B, Criterion II, which states that the QA program shall provide control over activities affecting the quality of the identified SSCs (i.e., those important to safety) to an extent consistent with their importance to safety. Thus, in accordance with Criterion II, controlling the software development activities should be applied consistent with the software's importance to safety.

Regulatory Guides (RGs) 1.168 through 1.173 contain some of the common industry requirements for software tool identification, configuration control, documentation, and V&V through endorsement of various IEEE standards. RG 1.168 endorses IEEE 1012-2004 for V&V requirements. RG 1.168 also endorses IEEE 1028-2008 for software reviews, inspections,

walkthroughs, and audits. IEEE 1028-2008 also contains an anomaly ranking and reporting scheme that should be considered by applicants and licensees when addressing the anomaly reporting documentation requirements of IEEE 829-2008. RG 1.169 endorses IEEE 828-2005 for planning and executing a SCM program. RG 1.170 endorses IEEE 829-2008 for documentation of software testing activities. RG 1.171 endorses IEEE 1008-1987 for software unit testing specification and documentation requirements. RG 1.172 endorses IEEE 830-1998 for requirements related to the development of a software requirements specification. RG 1.173 endorses IEEE 1074-2006 for defining a set of software life cycle processes and constituent activities appropriate for the development of safety system software.

RGs 1.168 through 1.173 only address a few of the software development life cycle processes and lack coverage for the project management, development, QA, integration, installation, maintenance, training, and operating processes. The existing software-related regulatory guides also lack guidance for software tool development and qualification and guidance on the use and qualification of COTS tools similar to that provided in RTCA DO-330, IEC 60880, ISO 26262, and BS EN 50128. Thus, the existing software-related regulatory guides are primarily written to address safety-related software development and do not address software tools beyond the simple high-level requirements. RGs 1.168 through 1.173 also lack guidance for software tool review and approval similar to the certification liaison process in RTCA DO-330.

ISG-06 provides guidance for the timing and scope of documentation that needs to be reviewed as part of an NRC review of license amendments supporting installation of DI&C equipment in nuclear power plants. ISG-06 recognizes two different methods of approving a component for use in safety-related applications based on whether the safety-related component critical characteristics can or cannot be verified. If a safety-related component has critical characteristics that can be verified, then the component may be designed and manufactured as a CGI and commercially dedicated. If the critical characteristics cannot be verified, then the component must be designed and manufactured under a 10 CFR Part 50, Appendix B-compliant QA program.

ISG-06 reiterates the requirements of IEEE 7-4.3.2-2003, Clause 5.3.2, that specify that tools must be developed to a similar standard as the safety-related software or used in a manner such that defects not detected by the tools are detected by V&V activities. ISG-06 also reiterates the requirements of BTP 7-14 which require the software development plan to identify tools and specify that tools should be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software developed using the tools. ISG-06 requires reviewers to evaluate tool usage in the overall context of the quality control and V&V processes to conclude that the tools function as described. Although the ISG-06 guidance is consistent with BTP 7-14 and accurately states the IEEE 7-4.3.2-2003 requirements, ISG-06 provides no additional guidance for the review of software tools other than the use of the tools may be audited to assess the quality of the tools.

BTP 7-14 provides the acceptance criteria for NRC review of planning documentation associated with the development of safety-related software. BTP 7-14 covers the required content of planning documentation of most software development life cycle processes similar to

the coverage of NASA NPR 7150.2A including project management, software development, software integration, software maintenance, software training, software operation, software testing, software safety, and support processes of verification, validation, QA, and CM.

The BTP 7-14 coverage of the content of planning documents is more extensive than RTCA DO-178C or RTCA DO-330 which only cover software development, verification, CM, and QA plans. BTP 7-14 acceptance criteria for planning documents includes many high-level requirements for software tools. Planning documents for most phases of the software development life cycle are required to describe tools used in that life cycle phase, define the process by which tools are selected, define the tool operational environment and any special controls necessary to execute the tools. In some cases, the planning document is required to contain a tool qualification plan.

The use of the term qualification in BTP 7-14 is unique to this document in that the general guidance of BTP 7-14 and other NRC documents is that software tools to be used in the development or testing of safety-related software, should themselves be developed as safety-related using a high-quality process equivalent to 10 CFR Part 50, Appendix B, or commercially-dedicated if the output of the tool cannot be proven to be correct. Consistent with 10 CFR Part 50, Appendix B, tools that are treated as safety-related, and hence must be qualified in accordance with BTP 7-14, must be done with a degree of rigor and level of detail appropriate to the safety significance of the software developed with the tools. However, BTP 7-14 does not provide or reference any specific guidance for qualifying the tool or developing the tool similar to RTCA DO-178C and RTCA DO-330, IEC 61508 and IEC 60880, ISO 26262, or BS EN 50128. Note that the use of the term “qualification” in BTP 7-14 for the process by which software tools are determined to be suitable and reliable for their application, and should not introduce flaws or fail to detect flaws in software that could impact safety, is inconsistent with the use of term qualification in the regulations, referring there to environmental and seismic qualification, which have no bearing on software or software tools. It appears that BTP 7-14 uses the term in a sense similar to that of RTCA DO-330. However, as explained previously, tool qualification as addressed in BTP- 7-14 and described in RTCA DO-330, could form a suitable basis for licensee or applicant acceptance of software tools for use with safety-related software, whether developed under a 10 CFR Part 50, Appendix B-compliant QA program or not, thus requiring acceptance by means of commercial-grade dedication.

3.4 Institute of Electrical and Electronics Engineers

NRC regulatory guides endorse several IEEE standards that contain requirements for promoting high functional reliability and establishing minimum functional requirements for computers used in nuclear power plants. Only a few IEEE standards discuss software tools and are limited to primarily V&V requirements. IEEE 1012-2012 and its predecessors, IEEE 1012-2004 and IEEE 1012-1998, discuss a four-level integrity scheme for software and software tools which can be tailored to fit many applications. However, IEEE 7-4.3.2-2010 and its predecessor IEEE 7-4.3.2-2003, and NRC regulatory guides do not endorse the use of all four integrity levels. NRC regulatory guides state that all safety-related software and nonsafety-related software executed on the same computer as safety-related software should all be classified as integrity level four

which requires the highest level of verification, validation, and QA. This is in contrast to most industries that fully utilize a three to five level schema, determine the safety significance of the software and software tool, and assign the software and software tool to one of those levels based on its safety significance or based on its probability of failure and the consequence of the failure.

IEEE 7-4.3.2-2010, its predecessor, IEEE 7-4.3.2-2003, and a proposed draft version contain high-level requirements for software and software tool verification, validation, QA, development, or dedication. IEEE 7-4.3.2-2003, IEEE 7-4.3.2-2010, and the draft version of IEEE 7-4.3.2 require that software V&V satisfy the requirements for the highest safety integrity level (i.e., SIL 4) as defined in IEEE 1012-1998, IEEE 1012-2004, and IEEE 1012-2012, respectively. The three standards also specify slightly different methods to determine the suitability of software tools to be used in the development of safety-related systems. Similar to the civil aviation industry, software and software tool development using a high-quality life cycle process or commercial-grade dedication is only required if the tool and its output cannot be verified and validated. IEEE 7-4.3.2-2003, IEEE 7-4.3.2-2010, and the draft version of IEEE 7-4.3.2 require software and software tool verification, validation, or dedication but lacks comprehensive verification, validation, or dedication processes, activities, or tasks comparable to the civil aviation industry (RTCA DO-178C and RTCA DO-330), the automotive industry (ISO 26262), the railway industry (BS EN 50128), or IEC standards (IEC 61508, IEC 61226, and IEC 60880).

The proposed draft version of IEEE 7-4.3.2 adds five tool categories: software development tools, surveillance and maintenance tools, development process support tools, in-line V&V tools, and off-line V&V tools. These five tool types are more detailed than the tool criteria used in the civil aviation industry, more detailed than the tool classes (i.e., non-classified, T1, T2, and T3) used in IEC 61508, and are similar to the five tool types (i.e., transformation tools, V&V tools, diagnostic tools, infrastructure tools, and configuration control tools) assigned in IEC 60880. The purpose of tool classes in all these applications is consistent; to determine the required level of verification and assessment.

IEEE 1012-2012 and its predecessor, IEEE 1012-2004, discuss V&V activities for many software development life cycle processes; however, the discussion of software tool V&V is limited to tools that insert or translate code and only covers QA aspects of tool use rather than actual methods and techniques for software tool V&V as discussed in IEC 60880, ISO 26262, and RTCA DO-330. Both IEEE 1012 versions also discuss V&V reporting requirements and content of the V&V plan. These requirements are similar to those of other industries and the documentation requirements of BTP 7-14. However, the two IEEE 1012 standards only contain requirements for the structure and content of the V&V plan associated with safety-related software development while BTP 7-14 covers all plans including the software management plan, software development plan, software implementation plan, software QA plan, software maintenance plan, software training plan, software operating plan, software configuration management (SCM) plan, and software test plan.

3.5 International Electrotechnical Commission

International Electrotechnical Commission (IEC) standards introduce the concept of functional safety, provide a risk-based approach for determining the required performance of safety-related systems, and provide generic requirements that can be used directly by industry or can be used as a basis for developing application-specific sector standards as is the case with BS EN 50128 (high-speed railway industry), ISO 26262 (automotive industry), and IEC 60880 (commercial nuclear power industry). IEC 61226 establishes a method of classification of the instrumentation and control (I&C) systems and equipment according to their importance to safety. IEC 60880 provides software requirements for nuclear power plant I&C systems performing category A functions.

IEC 61508 specifies a four-level software safety integrity scheme that is similar to schemes in BS EN 50128, ISO 26262, and RTCA DO-178C. Unlike RTCA DO-178C which uses a deterministic method of classifying software based on potential errors and the impact of those errors, IEC 61508 used a probabilistic risk-based methodology of assigning the software SIL. The probabilistic risk-based software SILs used in BS EN 50128 and ISO 26262 are based on the generic IEC 61508 standard. In all three standards (i.e., IEC 61508, BS EN 50128, and ISO 26262) the risk-based SIL determines the applicable software development requirements and determines the risk mitigation methods and the techniques and measures for meeting the applicable requirements at the desired probability. Unlike ISO 26262 and RTCA DO-178C which use the software safety classification or integrity level to determine software tool requirements, IEC 61508 is similar to BS EN 50128 in that the standard does not use the software SIL to determine the requirements for software tool qualification, selection, or use.

IEC 61508 contains all the common industry requirements with respect to adequate planning for tool use, tool selection, CM of tools, V&V of tools, and adequate documentation of tool use. The requirements apply to all software support tools. Similar to BS EN 50128, IEC 61508 defines tool classes (i.e., non-classified, T1, T2, and T3) similar to the three criteria used in RTCA DO-178C to classify tools. IEC 61508 tool class T3 covers software development tools similar to Criterion 1 in RTCA DO-178C. IEC 61508 tool class T2 covers software V&V tools similar to Criteria 2 and 3 in RTCA DO-178C. IEC 61508 tool class T1 has no equivalent in RTCA DO-178C and covers tools that cannot impact the executable code. Software tool requirements vary by tool class but not by the SIL of the software developed with the software tool. Although IEC 61508 requires some tools to be validated and identifies several optional methods of performing the validation, IEC 61508 lacks detailed software tool qualification requirements and qualification methods compared to RTCA DO-178C and RTCA DO-330.

IEC 60880 is a standard that addresses software aspects of computer-based safety-related systems that is a nuclear power industry application of IEC 61508. IEC 60880 contains general requirements consistent with the high-level requirements of IEC 61508 which requires identification, documentation, and CM of tools used in the software development process. IEC 60880 does not use the T1, T2, and T3 tool class designations defined in IEC 61508 to determine tool qualification requirements. Instead, IEC 60880 identifies five different tool types and addresses tool requirements based on the life cycle in which the tools are used. Tools

whose output is systematically verified, tools whose potential faults are mitigated, and tools that cannot introduce faults into the software are not required to be qualified. Tools that must be qualified must be developed and evaluated using the life cycle process of IEC 60880 and qualified using one or more of five alternative methods of qualification similar to the methods of qualification specified in RTCA DO-330. However, IEC 60880 does not provide guidance as to which of the five methods are required, recommended, or preferred based on the software classification or tool type. Unlike RTCA DO-330 which provides a comprehensive COTS tool qualification strategy, IEC 60880 does not specifically address COTS tool qualification.

3.6 Automotive Industry

Development and integration of new functionalities (i.e., collision avoidance and detection, in-vehicle dynamics control, active and passive safety systems) require safe system development processes and guidance to avoid risks from systematic failures through appropriate requirements and processes. ISO 26262 is an international standard that is an automotive specific implementation of IEC 61508 and provides requirements for the functional safety of road vehicles.

ISO 26262 uses a four-level, probabilistic risk-based software classification system (i.e., an automotive safety integrity level (ASIL)) that is virtually identical to the four-level safety integrity scheme used in IEC 61508. The ISO 26262 ASILs are also similar to the five-level safety integrity schemes used in BS EN 50128 and RTCA DO-178C. The ASIL determines the applicable requirements of ISO 26262 that need to be satisfied. The ASIL also determines which measures and techniques are recommended to ensure a sufficient and acceptable level of safety is achieved.

ISO 26262 provides broader coverage and a more in-depth process for confirming the safety of software tools compared to IEC 61508. With regard to coverage, IEC 61508 covers software tools that generate output that directly or indirectly contributes to the executable code of the safety-related system. ISO 26262 applies to any development process of a safety-related item or element beyond just the generation of the executable code including requirements specification, CM, and design support.

With respect to the more in-depth processes of ISO 26262, the assignment of a tool impact class, a tool error detection class, and determining the tool confidence level goes beyond the single assignment of a tool class in IEC 61508. The tool impact classes of TI1 and TI2 in ISO 26262 are comparable to the tool classes in IEC 61508; however, ISO 26262 does not distinguish between verification tools and development tools as does IEC 61508. The TI1 class of tools in ISO 26262 is similar to the T1 class of tools in IEC 61508. The TI2 class of tools in ISO 26262 is similar to the combined T2 and T3 classes of tools in IEC 61508. Where IEC 61508 just uses the tool class to determine the applicable software tool requirements, ISO 26262 adds a tool error detection class of TD1, TD2, and TD3 and a tool confidence level of TCL1, TCL2, and TCL3 to determine the qualification requirements of software tools. However, even though ISO 26262 contains a more in-depth process for determining the qualification requirements of software tools, it does not require qualification or specify requirements for software tools in the TCL1 class. As a minimum, the TCL1 class of software tools should be

verified and validated to ensure the tool performs its intended function and performs the function correctly even if the software tool has no impact on the safety-related item or element. With respect to the optional qualification method of development in accordance with a safety standard, ISO 26262 states the no safety standard is fully applicable to the development of software tools. However, RTCA DO-330 is a standard that is fully applicable to the development of software tools, therefore, this statement in ISO 26262 is no longer valid. Developing a software tool in accordance with the requirements of RTCA DO-330 using a tool qualification level (TQL) of TQL-1 should be acceptable for high-integrity safety-related software tools.

ISO 26262 contains minimal requirements for the content and structure of documentation for data generated during the software development life cycle processes, activities, and tasks or for planning the various aspects of the development process analogous to BTP 7-14. ISO 26262 lacks detailed requirements for documenting the activities performed by tools. ISO 26262 also lacks detailed review and approval practices similar to ISG-06 or the certification liaison processes discussed in RTCA DO-178C and RTCA DO-330. ISO 26262 also lacks a requirements compliance matrix to facilitate the software development and review and approval processes.

3.7 Railway Industry

Several European standards address the reliability, availability, maintainability, and safety for railway applications. BS EN 50128 is a British, railway specific implementation of IEC 61508 that defines processes based on a system life cycle for managing the safety of systems and components containing software.

BS EN 50128 specifies a five-level software safety integrity scheme that is similar to schemes in IEC 61508, ISO 26262, and RTCA DO-178C. However, in the current revision of BS EN 50128, only three of the five levels are effectively used. Unlike RTCA DO-178C which uses a deterministic method of classifying software based on potential errors and the impact of those errors, BS EN 50128 used a probabilistic risk-based methodology of assigning the software SIL. This risk-based SIL determines the applicable software development requirements of BS EN 50128 and determines the risk mitigation methods and the techniques and measures for meeting the applicable requirements at the desired probability. Unlike ISO 26262 and RTCA DO-178C which use the software safety classification or integrity level to determine software tool requirements, BS EN 50128 does not use the software SIL to determine the requirements for software tool qualification, selection, or use.

BS EN 50128 contains all the common industry requirements for software tool identification, configuration control, and documentation. The requirements apply to all software support tools. BS EN 50128 also defines tool classes (i.e., T1, T2, and T3) similar to the three criteria used in RTCA DO-178C to classify tools. BS EN 50128 tool class T3 covers software development tools similar to Criterion 1 in RTCA DO-178C. BS EN 50128 tool class T2 covers software V&V tools similar to Criteria 2 and 3 in RTCA DO-178C. BS EN 50128 tool class T1 has no equivalent in RTCA DO-178C and covers tools that cannot impact the executable code. Software tool requirements vary by tool class but not by the SIL of the software developed with

the software tool. Although BS EN 50128 requires some tools to be validated and identifies several optional methods of performing the validation, BS EN 50128 lacks detailed software tool qualification requirements and qualification methods compared to RTCA DO-178C and RTCA DO-330.

3.8 International Atomic Energy Agency

International Atomic Energy Agency (IAEA) publications include the IAEA safety standards including safety guides. IAEA NS-G-1.1 is an active safety guide that provides guidance for demonstrating the safety of software used in computer-based systems of commercial nuclear power plants. IAEA DS-431 is a draft safety guide intended to supersede IAEA NS-G-1.1.

IAEA NS-G-1.1 is similar to BTP 7-14 in that it focuses on providing guidance for preparing documentation in each software development life cycle phase to demonstrate the safety of software important to safety used in nuclear power plant computer-based systems. IAEA NS-G-1.1 contains all the high-level, common industry requirements for software tool identification, configuration control, and documentation. IAEA NS-G-1.1 also requires all software tools to be qualified to a level commensurate with their function in the development of the software and the correctness of the software tool output should be ensured by a tool certification process. Requiring software tool qualification based on its importance to safety is consistent with 10 CFR Part 50, Appendix B. Although IAEA NS-G-1.1 requires tool qualification and certification of the tool output, detailed tool qualification and certification processes, activities, and tasks comparable to RTCA DO-178C and RTCA DO-330 are not provided. IAEA NS-G-1.1 does not define software classifications but does define two tool categories (i.e., development tools and verification tools). General requirements of dependability and applicability vary for the two software tool categories but detailed requirements are lacking. IAEA NS-G-1.1 should expand the tool categories to cover all types of tools beyond development and verification tools and provide detailed qualification and certification requirements.

IAEA DS-431 maintains the similarity to BTP 7-14 by focusing on providing guidance for preparing documentation in each software development life cycle phase to demonstrate the safety of software important to safety used in nuclear power plant computer-based systems. IAEA DS-431 contains all the high-level, common industry requirements for software tool identification, configuration control, and documentation. IAEA DS-431 eliminates the two software tool categories of development and verification tools in favor of providing examples of software tools that are used throughout the software development life cycle. IAEA DS-431 requires tools to be verified and assessed consistent with tool reliability requirements. However, IAEA DS-431 does not supply any details on verification methods or assessment techniques. The high-level guidance of IAEA DS-431 needs to be supplemented with low-level requirements which tailor the requirements to the different types of tools.

Similar to IEC 61508, IEC 60880, and EPRI 106439, IAEA NS-G-1.1 and IAEA DS-431 identify a concern for common mode failures in software. Even though the benefits of countermeasures are difficult to estimate, IAEA NS-G-1.1 and IAEA DS-431 stress the need for independence, diversity, and a comprehensive qualification strategy to protect against common mode failures.

3.9 Electric Power Research Institute

The Electric Power Research Institute (EPRI) produces thousands of products annually, ranging from white papers and newsletters to comprehensive reports. Some EPRI reports complement NRC regulatory guides by providing guidance for meeting the commercial-grade dedication requirements of 10 CFR Part 21 and the QA requirements of 10 CFR Part 50, Appendix B. EPRI NP-5652, EPRI TR-102260 focus on commercial-grade dedication of hardware components. EPRI TR-106439 and EPRI TR-107339 provide guidance for accepting commercial-grade digital equipment. EPRI TR-1025243 provides guidance for the commercial-grade dedication of safety-related design and analysis software tools. The recommended approach to commercial-grade dedication in these documents could be expanded and adapted as a method to qualify all software tools.

EPRI guidance for commercial-grade dedication of CGIs is to follow a generic process that consists of the main steps of a safety evaluation, identification of critical characteristics, and acceptance. Similar to the aerospace industry's safety litmus test, the commercial-grade dedication process performs a safety evaluation to determine whether an item is safety-related or nonsafety-related. Only safety-related items need to be dedicated. Determining the critical characteristics is similar to the identification and assignment of a tool confidence level in ISO 26262. The commercial-grade dedication acceptance process consists of a combination of four alternative methods which have similarities in other industries. Method 1 is special tests and inspections which is similar to the validation method specified in IEC 61508, ISO 26262, and BS EN 50128 or exhaustive input testing used in the civil aviation industry. Method 2 is a survey of the supplier's QA program which is also an alternative method in ISO 26262. Method 3 is source verification and is similar to the practice in other industries of auditing the QA practices of the supplier. This method is typically not viable for COTS software tools since it must be performed during the computer program development life cycle. Method 4 is reliance on the item's performance record which is similar to the practice in other industries of using past performance and designing software for reuse.

3.10 National Institute of Standards and Technology

National Institute of Standards and Technology (NIST) Computer Systems Laboratory (CSL) develops standards and guidelines for use by agencies and organizations in industry, government, and academia. NIST special publication (SP) 500-234 addresses software V&V of computing systems employed in the health care environment. NIST SP 500-234 is very similar to IEEE 1012-2012. Both documents discuss required V&V activities and recognize that software V&V is tightly integrated with the software development process. Both NIST SP 500-234 and IEEE 1012-2012 recommend that an independent V&V (IV&V) team execute qualification tests on tools (e.g., compilers, assemblers, and utilities) shared with the development environment and recommend that the IV&V team should use or develop its own set of test and analysis tools when possible. Both NIST SP 500-234 and IEEE 1012-2012 also state that the V&V plan should include details for acquiring tools and for training personnel. However, NIST SP 500-234 does not discuss software integrity levels or provide minimum V&V activities as a function of integrity level. NIST SP 500-234 does not distinguish between different categories of tools or use safety as a measure to determine tool qualification

requirements. NIST SP 500-234 also does not provide guidance for developing the required independent software tools or provide a process for qualifying software tools. NIST SP 500-234 does discuss a need for artificial intelligence techniques and knowledge-based systems to manage the vast knowledge in the health care industry. However, these advanced computer based systems have no current application in the commercial nuclear power industry. NIST SP 500-234 does not provide any additional software tool requirements beyond the requirements that already exist in IEEE 1012-2012.

3.11 Atomic Energy of Canada Limited

The Atomic Energy of Canada Limited (AECL) maintains a library of published information including books, journals, standards, and technical reports developed through cooperation with Canadian utilities. One standard, AECL CE-1001-STD, specifies requirements for safety critical (Category I) software engineering for Canadian-invented Deuterium-Uranium (CANDU®) nuclear generating stations. The AECL standard supports a three-level software classification system where Category I software requires more rigorous verification, hazard analysis, and qualification processes than Category II or III software. AECL CE-1001-STD also describes a minimum set of software engineering processes that are required for safety critical software development. The standard contains few requirements for software tools and those requirements focus on the high-level requirements of identifying tools, documenting tool use, and planning budgets and resources for tool development and training. AECL CE-1001-STD does not distinguish between different types of software tools, does not use software tool categories or classifications, and does not contain any specific software tool qualification requirements tailored to the type of tool. The coverage of high-level requirements in AECL CE-1001-STD is similar to the high-level requirements of FAA-STD-026A but lack the detailed documentation requirements of FAA-STD-026A.

Intentionally Blank

4 Software Tool Hazard Assessment

According to the Multinational Design Evaluation Program (MDEP) Digital Instrumentation and Control Working Group (DICWG) Generic Common Position DICWG No 2, the use of appropriate software tools can increase the integrity of the instrumentation and control (I&C) development process and the software reliability by reducing the risk of introducing faults during the process. The use of tools can also have economic benefits by reducing the time and effort required to produce systems, components, and software. Tools can be used to automatically check for adherence to standards, generate records and documentation, and support configuration management (CM). Tools are primarily used to capture system and software requirements; transform requirements into final system code and data; perform verification, validation, and testing; prepare and control application data; and manage and control processes. Tools can also reduce the effort required for testing and maintaining logs. Tools are most powerful when they are clearly defined and designed to work co-operatively with each other. Although tool-automated and tool-assisted engineering activities may result in fewer errors than manual engineering activities if tools are well designed, carefully developed, and appropriately used, the undetected faults in automated tools or tool-assisted engineering activities used in developing software for the I&C system of a commercial nuclear power plant may still pose a risk to the nuclear power plant, to the people, and to the environment.

Computerized tools used in the development of software are referred to as software tools or computer-aided software engineering (CASE) tools. These tools can be used at almost every phase of the software life cycle and assist in almost every task. The tools may be individual programs or they may be combined into a single program that performs multiple related tasks such as an integrated development environment (IDE).

This section provides an assessment of the hazards associated with the use of software tools in different stages of the I&C system life cycle. A hazard is a potential source of harm (i.e., physical injury or damage to the health of people or damage to property or the environment). Tools can be a hazard when defects in software tools introduce or fail to detect defects in safety-related I&C software causing the safety-related software to fail and not perform its intended safety-related function. IEEE 7-4.3.2-2010, Annex D, describes a method of identifying hazards that can occur when using software. This guidance states that only those conditions with significant consequences and significant probability of occurrence need to be resolved. The capability maturity model integration (CMMI) risk management process requires the identification of potential problems before they occur so that risk handling activities can be planned and invoked as needed. This process has three goals.

1. Prepare by determining risk sources
2. Identify and document risks using defined risk categories
3. Develop a risk mitigation strategy

In IEC 61508, the second step of the analysis process is a hazard analysis/risk assessment with the goal of determining the required safety integrity level (SIL) of the software to be developed.

The SIL review identifies all the process hazards, estimates the risk of failure, and tries to determine that, if a failure occurs, the product will “fail safely”.

The common theme of all of these is that the results of a failure should be taken into account and mitigation strategies should be planned. If a failure in a tool cannot affect any safety sensitive system then the tool has no need to be regulated. Even when the tool is used to develop a safety system, failures that have no or little effect on the safety of the product should be recognized and documented, but need not have a planned mitigation strategy.

Sections 4.1 through 4.9 provide a hazard assessment of software tools based on their use as described in the nine generic computer-aided software engineering (CASE) tool categories. The nine categories of tools were generated by Information Systems Laboratories, Inc. (ISL) based on the CASE tool characteristics and attributes discussed in IEEE 14102-2010 which is an update of IEEE 1462-1998. Additional information from IEEE 12207-2008 and IEEE 1074-1995 was used by ISL to determine where the tools would be used in the life cycle process. A thorough discussion of IEEE 12207-2008 and IEEE 1462-1998 is contained in Section 2 of ISL-ESD-TR-13-05 which is the Task 1 report from this research project.

By combining the CASE tool attributes in IEEE 14102-2010 with the life cycle processes in IEEE 12207-2008, ISL has generated nine tool categories. Tools associated with verification and validation (V&V) processes have been combined into a single category since these processes are closely related. Since the science of software engineering changes so rapidly, the nine categories are not meant to be an all-inclusive listing but rather a guideline to classify software tools that may be used in development of safety-significant software. Other IEEE 12207-2008 life cycle processes not addressed by these nine tool categories are absent either because CASE tools do not typically provide support for those processes, or because the process and/or the CASE support for it are not stable at this time.

4.1 Management Support Tools

Management support tools are software tools that can be found in almost any type of business and are not specific to software development. These tools are generally classified as resource planning tools and do not directly affect the final I&C software. Therefore, they have no impact on the safety of any systems developed using them. An ISL observation, based on the fact that these tools function at the project level and do not directly affect the final software, is that no regulatory guidance is necessary for these tools. However the quality of the tools does impact the project risk as scheduling and resource planning are important to the success of a project. Errors in the use of the tool or in the tool itself may cause project schedule delays or cost overruns even though they will not have a direct effect on the quality of the software product. This should be recognized when the project risk analysis is performed.

4.2 Implementation Tools

Implementation tools are used in the design process to support elicitation, analysis, specification, verification, validation, and management of software requirements. This includes tracing requirements in the code and the test cases.

Requirements management is the process of documenting, analyzing, tracing, and prioritizing requirements throughout the software life cycle. The requirements gathering process is essential and significant productivity gains can be realized if this process is improved. In a study of failures involving industrial control systems done by the Health and Safety Executive in the United Kingdom, 44% of failures were caused by incorrect or incomplete specifications. Requirements management is a continuous process that includes bi-directional tracing of requirements to code. RG 1.172 addresses the software requirements gathering process. It specifies that requirements should exhibit characteristics such as correctness and completeness. RG 1.172 specifies the following characteristics of the software requirements process.

- (1) Traceability and Accuracy
- (2) Completeness
- (3) Consistency
- (4) Ranking for Importance and Stability
- (5) Verifiability
- (6) Modifiability

A software tool used for requirements gathering can address any or all these areas. Possible failure in this type of software tool could include:

- (1) Loss of requirements
- (2) Incorrect priorities
- (3) Failure to keep or define timing of requirements
- (4) Loss or incorrect mapping to the code
- (5) Loss or incorrect mapping to a test case

As part of the installation of any requirements tool a test case should be performed to check all of these possibilities.

There is a strong two way connection between the requirements gathering process and the quality assurance (QA) process. Every requirement should be represented somewhere in the product test cases. The CMMI process area Requirements Management (REQM) specific goal SP 1.5 requires the identification of inconsistencies between project work and requirements. Inconsistencies in the requirements as generated by a tool should then be noticed during the QA process. Therefore, although this may be a project risk because of schedule delays, it is a low safety risk when used in a well-defined software environment.

An ISL observation, based on the fact that the output of some of these tools can be used directly in the generation of source code and databases, is that implementation tools should

have regulatory guidance. The extent of the regulatory guidance and review and approval process should be based on the effect that the tool can have on the final software.

4.3 Modeling Tools

Modeling tools are used in the design process to simplify development of the software architecture, database design, user interface design, and other tasks. Software modeling tools can be used to create generic representations of the software that are independent of a programming language. The risk associated with failures of a modeling tool is dependent on whether the tool can be used to generate code either as a prototype or as all or part of the final software product. Any tool that can be used to prototype can produce code that might end up in the finished product. If the modeling tool cannot or is not ever used to generate code, then the associated risk is low. In this case the modeling tools can be considered and in the same category as the process implementation tools and a low safety risk.

However, when a tool can and is used to generate code, even as a prototype, it should be subject to a higher level of scrutiny. In particular there is a class of tools used for Model Driven Development (MDD) (also known as Model Driven Engineering) that fits under this definition and directly generates code. MDD is a method of software development where the primary focus and products are models rather than computer programs. The major advantage of this is that the models are expressed using concepts that are less bound to the underlying implementation technology and are closer to the problem domain relative to most popular programming languages.

Model driven development uses languages such as XML and UML. These tools allow the models to be formally verified. However, Selic states that:

“If an error is detected in the generated program, finding the place in the model that must be fixed either at compile time or runtime might be difficult. In traditional programming languages, we expect compilers to report errors in terms of the original source code and, for runtime errors, we now expect a similar capability from our debuggers. The need for such facilities for models is even greater because the semantic gap between the modeling language’s high-level abstractions and the implementation code is wider. This means that model-level error reporting and debugging facilities (in essence, “decompilers”) must accompany practical automatic code generators. Otherwise, the practical difficulties encountered in diagnosing problems could be significant enough to nullify much of MDD’s advantage. Programmers faced with fixing code that they don’t understand will easily break it and will likely be discouraged from relying on models in the future.”

While automated code generators have the potential to eliminate manual coding errors as well as to reduce costs and development times, the generated code needs to be proven to be correct and complete. Errors from this type of software tool could have serious consequences if undetected. A standard set of problems should be executed when the code generator is installed. But, if the code generator is not qualified for safety-related use, it will be necessary to test and certify all the resultant code as if it were written manually.

4.4 Construction Tools

Construction tools are used in the development process to translate source code into machine executable code, create databases, and user interfaces. This category includes all the most commonly used software development tools. Examples include:

- (1) makefiles
- (2) compilers and optimizing compilers
- (3) debuggers
- (4) linkers
- (5) integrated development environments (IDEs)
- (6) database development tools
- (7) user interface and web page development tools

Many of these tools are very complex with a large code base. It is generally concluded that formal verification of compilers, particularly optimizing compilers, is not feasible. Zuck states that:

“Formally verifying an optimizing compiler, as one would verify any other large program, is not feasible due to its size, ongoing evolution and modification, and, possibly, proprietary considerations.”

This does not mean that optimization should be precluded. Rather, that the optimized version of the software should be validated against a version that is not optimized to ensure that there is no change in code behavior.

There is a lot of research on the subject of compiler V&V available, particularly in the area of increasing correctness and security. Hall, Padua, and Pingalli state that:

“Although the compiler field has transformed the landscape of computing, important compilation problems remain, even as new challenges (such as multi-core programming) have appeared. The unsolved compiler challenges (such as how to raise the abstraction level of parallel programming, develop secure and robust software, and verify the entire software stack) are of great practical importance and rank among the most intellectually challenging problems in computer science today.”

Construction tools can come from both vendors and open source groups. It is unlikely that vendors would wish to submit to an inspection because of the proprietary nature of the code source. Open source, by its very nature, could be examined but the task would be enormous and, because of the frequent changes to the code basis, probably neither useful nor cost effective. Some vendors put a disclaimer in their license agreements rather than to suggest that their product is certified for safety-related use. For example, Oracle states that:

“The Software is developed for general use in a variety of information management applications; it is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury.”

When development tools are used to develop safety-related software, flaws in the tool could affect the quality of the software. Unfortunately the types of failure are varied and depend not only on the type of development tool used but also on the options that are chosen, the surrounding process, and on the other tools in tool chain. For example, a “make” tool may fail to detect all the modules that should be recompiled. But if a clean build is always mandated, then the process, in which the tool is used, precludes this possible failure mode. Failures which do not create a working executable can also be excluded as they have no impact on safety.

4.5 Maintenance Tools

Maintenance tools support troubleshooting, fault isolation, reverse engineering, and other similar tasks. These tools can have the ability to alter the software in the I&C systems and may fail to identify faults during use. All the tools described in the development tools section can also be used as maintenance tools. This category also includes reverse engineering and restructuring tools, which include decompilers and disassemblers. Sometimes automated documentation tools are also included in this category, but for the purposes of this discussion those will be in considered in the documentation section.

Decompilers and disassemblers are used to examine or change legacy or third party code where the source code is no longer or never was available. There is a great risk in using code obtained in this manner. It is unlikely to be able to be tied to requirements and test cases. The code obtained is usually cryptic and difficult to read. While the information obtained from reverse engineering can be useful for understanding how an existing product functions, code obtained from reverse engineering tools should not be used in developing safety-related software.

4.6 Documentation Tools

Tools that are used to support the documentation process include text editors, graphical editors, and web design tools. Testing tools may also automate the documentation of test results and compliance with test requirements. Although clear documentation is a necessary part of the software development process, tools used only to produce documentation have no direct effect on the final software output. Documentation tools are likely to reduce risk by making it easier to keep documentation available and up to date.

4.7 Configuration Management Tools

CM tools can have a direct effect on the final software output by allowing unauthorized or untracked modifications to be made to the software. These tools control access to the source code, track changes, and may be used to generate the build package for compilation. Software products for CM are designed to control a repository of components that are combined to produce a software product and all of its versions. A short summary of features and risks include:

- (1) Versioning – the correct component set must be selected for compilation and linking for multiple versions of a product
- (2) Cooperative work support. –if more than one developer is working on a modules, changes must be synchronized and the merges managed
- (3) Backups – prior versions of code modules should be able to be recalled

CM tools should at the minimum be tested and verified to give correct results in these areas within the framework of a well-defined process.

4.8 Quality Assurance Tools

QA tools are used to support user entry of quality data, analyze quality data, and to support risk identification, estimation, and impact. None of these tools has a direct effect on the final software output. They may require regulatory guidance because they are used as a part of a 10 CFR Part 50, Appendix B, program, but the use of such tools to analyze and track quality and risk is likely to reduce risk for the project as a whole.

4.9 Verification and Validation Tools

V&V tools are used to simplify and organize software testing. They may or may not have a direct effect on the final software output depending on the tool characteristics. As discussed in the proposed draft of IEEE 7-4.3.2, some V&V tools (i.e., in-line V&V tools) may have a direct effect on the final software output because the safety-related software or logic can be automatically or manually changed by the tool user based on the results of the V&V process. All V&V tools have the capability of failing to locate defects in the software or they may be used to justify eliminating other V&V tasks or to justify eliminating programming functions that can catch faults such as fault handling subroutines. V&V tools are used to evaluate the quality of the software system. Verification is the process of checking whether the software system meets the specified requirements while validation is the process of checking whether the system meets the actual requirements. V&V tools are used throughout the software life cycle and include both tools that use formal methods as well as tools for designing, storing, and running automated test cases. For both types of tools, there must be links to the requirements.

The potential hazards from these two types of tool are quite different. Formal verification tools are used to show that the software, as built, is correct and, perhaps, to justify reducing the testing requirements. Testing tools take the other approach and generally design and execute a test case for each verifiable requirement. The success of either method is strongly dependent on the quality of the requirements.

Formal verification has limits because it may be difficult to do within time and budget constraints of a project and because there are elements that are not amenable to formal validation such as the usability of a graphical interface. On the other hand, the value of simply running test cases depends on the design of the test cases and the data that is used. Using a combination of these two methods is generally considered to give the best results and to provide some measure of checks and balances.

4.10 Conclusions

While it is unlikely that any tool can be certified as completely correct and error free, the use of a software development tool is generally considered to produce a better product than manual methods. Many tools have a large user base which provides feedback, are well supported, and have been in use long enough to be well tested. The risk of failures caused by using such a tool is less than the risk of failure from not using any tool.

It is also important to consider all the tools used and their place in the software process. Risk can be reduced by an appropriate selection of tool options and failures can be detected or mitigated by the use of tool combinations.

5 Baselines

This section contains baseline requirements, guidance, and review and approval practices. The baselines in the Task 1 report have been analyzed by collectively examining all the baselines, separating them into constituent elements, and finding the essential features. The baselines have been reorganized by tool specific activity rather than by software development life cycle because most software tool requirements and guidance deal with planning for tool use, selecting tools, documenting tool use, maintaining tools under configuration management (CM), and qualifying tools which apply to more than one software development life cycle. The reorganization of the baselines results in a better understanding of the type and applicability of the tool requirements, guidance, and review and approval practice. The reorganized baseline requirements have also been compared with baseline guidance to establish the relationships. Some of the baselines, originally written in the language used by the industry or organization, have also been rewritten to use NRC specific terminology in preparation for Task 3 which will provide the technical basis for new or improved NRC regulatory guidance.

5.1 Baseline Additions

As part of the baseline analysis and using the results of the document analysis from Section 2, new baselines have been added where new requirements, guidance, or review and approval practices were discovered during the analysis of new documents or previously surveyed documents. New baseline requirements are identified in Section 5.1.1 and new guidance, review, and approval practices are identified in Section 5.1.2. These new baselines have been consolidated into the final baselines in Section 5.3.

5.1.1 Baseline Requirements Additions

5.1.1.1 *Software and Software Tool Classifications*

1. Functions to be performed by instrumentation and control (I&C) systems shall be assigned to categories according to their importance to safety that is identified by means of the consequences in the event of its failure when it is required to be performed and the consequences in the event of a spurious actuation. {IEC 61226, Section 5.1}
2. A software on-line support tool shall be considered to be a software element of the safety-related system. {IEC 61508-3, Section 7.4.4.1}
3. Software off-line support tools are those that support a phase of the software development life cycle, cannot directly influence the safety-related system during its run time. (IEC 61508-4, Section 3.2.11)
4. The intended usage of the software tool shall be analyzed and evaluated to determine the tool impact (TI) class and the tool error detection (TD) class. The TI class and the TD class are used to determine the tool confidence level (TCL). {ISO 26262-8, Sections 11.4.5.2 through 11.4.5.5}

- a. The TI class is a measure of the possibility that a malfunction of a particular software tool can introduce or fail to detect errors in a safety-related item or element being developed.
- i. Class TI1 shall be selected when there is an argument that there is no such possibility.
 - ii. Class TI2 shall be selected in all other cases.
- b. The TD class is an expression of the confidence in measures that prevent the software tool from malfunctioning and producing corresponding erroneous output, or in measures that detect that the software tool has malfunctioned and has produced corresponding erroneous output. If a software tool is used for the tailoring of the development process in such a way that activities or tasks are omitted, TD2 shall not be selected.
- i. Class TD1 shall be selected if there is a high degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected.
 - ii. Class TD2 shall be selected if there is a medium degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected.
 - iii. Class TD3 shall be selected in all other cases
- c. The required software TCL shall be determined according to Table 5.4 based on the TI and TD classes.

Table 5.1 – Determination of the Tool Confidence Level (TCL)

		Tool error detection		
		TD1	TD2	TD3
Tool impact	TI1	TCL1	TCL1	TCL1
	TI2	TCL1	TCL2	TCL3

5.1.1.2 Software Tool Planning

1. An overall safety validation plan shall be developed that shall include the required environment in which the validation activities are to take place (for example, for tests this would include calibrated tools and equipment). {IEC 61508-1, Section 7.8.2.1.h}
2. Planning for the validation of a safety-related system shall consider the required environment in which the testing is to take place including all necessary tools and

equipment (also plan which tools and equipment should be calibrated). {IEC 61508-2, Section 7.3.2.2.d}

3. Safety-related system verification planning shall be carried out for each phase of the safety life cycle and address all the criteria, techniques, and tools to be utilized in the verification for that phase. {IEC 61508-2, Section 7.9.2.2; ISO 26262-8, Section 9.4.1.1.e}
4. Methods, measures, and tools for quality assurance (QA) according to the allocated safety integrity levels (SILs) shall be specified or referenced in the software QA plan. {BS EN 50128, Section 6.5.4.5.e}
5. The validation plan for software aspects of system safety shall consider the required environment in which the validation activities are to take place (for example, for tests this could include calibrated tools and equipment). {IEC 61508-3, Section 7.3.2.2.g}
6. The software verification planning shall refer to the criteria, techniques, and tools to be used in the verification activities, and shall address the selection and utilization of verification tools (test harness, special test software, input/output simulators, etc.) {IEC 61508-3, Section 7.9.2.2.c}
7. The usage of a software tool shall be planned, including the determination of: {ISO 26262-8, Section 11.4.4.1}
 - a. the identification and version number of the software tool;
 - b. the configuration of the software tool;
 - c. the use cases of the software tool;
 - d. the environment in which the software tool is executed;
 - e. the maximum SIL of all the safety requirements, allocated to the item or element that can be violated, if the software tool is malfunctioning and producing corresponding erroneous output; and
 - f. the methods to qualify the software tool, if required based on the determined level of confidence.

5.1.1.3 *Software Tool Documentation*

1. Software testing shall be documented by a test specification that shall document the test environment, tools, configuration, and programs used for testing. {BS EN 50128, Section 6.1.4.4.d}
2. The information documented during the overall safety validation process shall include tools and equipment used, along with calibration data. {IEC 61508-1, Section 7.14.2.3}

-
3. The verification plan shall document or refer to the criteria, techniques, and tools to be used in the verification activities. {IEC 61508-1, Section 7.18.2.2; BS EN 50128, Section 6.2.4.5}
 4. The output of the functional safety assessment of a compliant item shall include the procedures, methods, and tools used for assessing the systematic capability along with the justification of its effectiveness to facilitate the re-use of the assessment results in the context of a larger system. {IEC 61508-1, Section 8.2.13.d}
 5. During the design of safety-related systems, system integration tests shall be planned and documentation of the test planning shall include the test environment, tools, configuration, and programs. {IEC 61508-2, Section 7.4.6.5}
 6. The safety-related system integration testing shall document the tools and equipment used along with calibration data. {IEC 61508-2, Section 7.5.2.6.d}
 7. The safety-related system operation and maintenance procedures shall specify the tools necessary for maintenance and revalidation and procedures for maintaining the tools and equipment. {IEC 61508-2, Section 7.6.2.1.g}
 8. Appropriate documentation of the safety-related system validation testing shall be produced which shall state for each safety function the tools and equipment used, along with calibration data. {IEC 61508-2, Section 7.7.2.4.c}
 9. The evaluation of the verification results shall contain the configuration of the verification environment and verification tools used, and the calibration data used during the evaluation, if applicable. {ISO 26262-8, Section 9.4.3.2.c}
 10. The software integration test specification shall state the test environment, tools, configuration, and programs {IEC 61508-3, Section 7.4.8.2.d; BS EN 50128, Section 7.3.4.37}
 11. The software/programmable electronics integration test specification (hardware and software) shall state the test environment including tools, support software, and configuration description. {IEC 61508-3, Section 7.5.2.2.d}
 12. For each safety function, software safety validation shall document the tools and equipment used together with calibration data. {IEC 61508-3, Section 7.7.2.5.d}
 13. The software architecture specification shall justify that the techniques, measures, and tools chosen form a set which satisfies the software requirements specification at the required software SIL. {BS EN 50128, Section 7.3.4.12}
 14. The following information shall be available to ensure the proper evaluation or usage of the software tool: {ISO 26262-8, Section 11.4.4.2}
 - a. a description of the features, functions, and technical properties of the software tool;
-

-
- b. the user manual or other usage guides, if applicable;
 - c. a description of the environment required for its operation;
 - d. a description of the expected behavior of the software tool under anomalous operating conditions, if applicable;
 - e. a description of known software tool malfunctions and the appropriate safeguards, avoidance, or work-around measures, if applicable; and
 - f. the measures for the detection of malfunctions and the corresponding erroneous output of the software tool identified during the determination of the required level of confidence for this software tool.
15. The description of the usage of a software tool shall contain the following information: {ISO 26262-8, Section 11.4.5.1}
- a. the intended purpose;
 - b. the inputs and expected outputs; and
 - c. the environmental and functional constraints, if applicable.
16. The qualification of the software tool shall be documented including the following: {ISO 26262-8, Section 11.4.6.2}
- a. the unique identification and version number of the software tool;
 - b. the maximum tool confidence level (TCL) for which the software tool is classified together with a reference to its evaluation analysis;
 - c. the pre-determined maximum ASIL, or specific ASIL, of any safety requirement which might be violated if the software tool is malfunctioning and produces corresponding erroneous output;
 - d. the configuration and environment for which the software tool is qualified;
 - e. the person or organization who carried out the qualification;
 - f. the methods applied for its qualification;
 - g. the results of the measures applied to qualify the software tool; and
 - h. the usage constraints and malfunctions identified during the qualification, if applicable.

5.1.1.4 Software Tool Selection and Use

1. Where applicable, automatic testing tools and integrated development tools shall be used to avoid systematic faults. {IEC 61508-2, Section 7.4.6.4}

2. Modifications to safety-related systems or software maintenance shall be performed with at least the same level of expertise, automated tools, planning, and management as the initial development of the safety-related system or software. (IEC 61508-2, Section 7.8.2.3; BS EN 50128, Section 9.2.4.17)
3. Software off-line support tools shall be selected as a coherent part of the software development activities. {IEC 61508-3, Section 7.4.4.2; BS EN 50128, Section 6.7.4.1}
4. The selection of the off-line support tools in classes T2 and T3 shall be justified. {IEC 61508-3, Section 7.4.4.4; BS EN 50128, Section 6.7.4.2}
5. All off-line support tools in classes T2 and T3 shall have a specification or product documentation that clearly defines the behavior of the tool and any instructions or constraints on its use. {IEC 61508-3, Section 7.4.4.4; BS EN 50128, Section 6.7.4.3}
6. The software development process for the software item, including life cycle phases, methods, languages, and tools, shall be consistent across all the sub-phases of the software life cycle and be compatible with the system and hardware development phases, such that the required data can be transformed correctly. {ISO 26262-6, Section 5.4.4}

5.1.1.5 **Software Tool Development, Qualification, and Dedication**

1. The compatibility of the tools of an integrated toolset shall be verified. {IEC 61508-3, Section 7.4.4.9}
2. Each new version of an off-line support tool shall be qualified. {IEC 61508-3, Section 7.4.4.18; BS EN 50128, Section 6.7.4.11}
3. The methods listed in Table 5.2 shall be applied for the qualification of software tools classified at TCL3. The methods listed in Table 5.3 shall be applied for the qualification of software tools classified at TCL2. A software tool classified at TCL1 needs no qualification methods. A table entry of (++) means the method is highly recommended while an entry of (+) means the method is recommended. The methods are either recommended or highly recommended depending on the automotive safety integrity level (ASIL) of the software being developed. {ISO 26262-8, Section 11.4.6.1}

Table 5.2 – Qualification of Software Tools Classified TCL3

Methods		ASIL			
		A	B	C	D
1a	Increased confidence from use	++	++	+	+
1b	Evaluation of the tool development process	++	++	+	+
1c	Validation of the software tool	+	+	++	++
1d	Development in accordance with a safety standard	+	+	++	++

Table 5.3 – Qualification of Software Tools Classified TCL2

Methods		ASIL			
		A	B	C	D
1a	Increased confidence from use	++	++	++	+
1b	Evaluation of the tool development process	++	++	++	+
1c	Validation of the software tool	+	+	+	++
1d	Development in accordance with a safety standard	+	+	+	++

- a. If a software tool is to be qualified using the method of increased confidence from use, then the following requirements shall be met. {ISO 26262-8, Section 11.4.7}
- i. Evidence is provided for the following:
 - (a) the software tool has been used previously for the same purpose with comparable use cases and with a comparable determined operating environment and with similar functional constraints,
 - (b) the justification for increased confidence from use is based on sufficient and adequate data which can be obtained through accumulated amount of usage (e.g., duration or frequency),
 - (c) the specification of the software tool is unchanged, and
 - (d) the occurrence of malfunctions and corresponding erroneous outputs of the software tool acquired during previous developments are accumulated in a systematic way.
 - ii. The experience from the previous usage of the software tool during given development activities shall be analyzed and evaluated by considering the following information:
 - (a) the unique identification and version number of the software tool;
 - (b) the configuration of the software tool;
 - (c) the details of the period of use and relevant data on its use;
 - (d) the documentation of malfunctions and corresponding erroneous outputs of the software tool with details of the conditions leading to them;
 - (e) the list of previous versions monitored, listing the malfunctions fixed in each relevant version; and

- (f) the safeguards, avoidance measures, or work-arounds for the known malfunctions, or detection measures for a corresponding erroneous output, if applicable.
 - iii. The increased confidence from use argument shall only be valid for the considered version of the software tool.
 - b. If a software tool is to be qualified using the method of evaluation of the tool development process, then the following requirements shall be met. {ISO 26262-8, Section 11.4.8}
 - i. The development process applied for the development of the software tool shall comply with an appropriate standard.
 - ii. The evaluation of the development process applied for the development of the software tool shall be provided by an assessment of the development of an adequate and relevant subset of the features of the software tool based on an appropriate national or international standard and the proper application of the assessed development process shall be demonstrated.
 - c. If a software tool is to be qualified using the method of validation of the software tool, then the following requirements shall be met. {ISO 26262-8, Section 11.4.9}
 - i. The validation of the software tool shall meet the following criteria:
 - (a) the validation measures shall demonstrate that the software tool complies with its specified requirements,
 - (b) the malfunctions and their corresponding erroneous outputs of the software tool occurring during validation shall be analyzed together with information on their possible consequences and with measures to avoid or detect them, and
 - (c) the reaction of the software tool to anomalous operating conditions shall be examined (e.g., foreseeable misuse, incomplete input data, incomplete update of the software tool, use of prohibited combinations of configuration settings).

5.1.1.6 Software Tool Quality Assurance

1. Measurement equipment used for testing shall be calibrated appropriately and any tools used for testing shall be shown to be suitable for the purpose. {BS EN 50128, Section 6.1.4.1}
 2. An assessment shall be carried out for offline tools in classes T2 and T3 to determine the level of reliance placed on the tools, and the potential failure mechanisms of the tools that
-

may affect the executable software. Where such failure mechanisms are identified, appropriate mitigation measures shall be taken (e.g., avoid known bugs, restrict use of the tool functionality, check the tool output, use diverse tools for the same purpose). (IEC 61508-3, Section 7.4.4.5; BS EN 50128, Section 6.7.4.2)

3. For each tool in class T3, evidence shall be available that the output of the tool conforms to its specification or documentation of the output or failures in the output are detected. {IEC 61508-3, Section 7.4.4.6; BS EN 50128, Section 6.7.4.4}
4. Where the results of tool validation are unavailable, there shall be effective measures to control failures of the executable safety-related system that result from faults that are attributable to the tool (e.g., generation of diverse redundant code which allows the detection and control of failures of the executable safety-related system as a result of faults that have been introduced into the executable safety-related system by a translator). {IEC 61508-3, Section 7.4.4.8; BS EN 50128, Section 6.7.4.6}
5. To the extent required by the SIL, the software or design representation selected shall have a translator which has been assessed for purpose including, where appropriate, assessment against the international or national standards. {IEC 61508-3, Section 7.4.4.10; BS EN 50128, Section 6.7.4.7}
6. Where automatic code generation or similar automatic translation takes place, the suitability of the automatic translator for safety-related system development shall be assessed at the point in the development life cycle where development support tools are selected. {IEC 61508-3, Section 7.4.4.14; BS EN 50128, Section 6.7.4.9}
7. Safety-related software jointly developed with a supplier requires a development interface agreement that specifies the supporting processes and tools, including interfaces, to assure a development environment compatible with the supplier. {ISO 26262-8, Section 5.4.3.1.g}
8. Software tool usage, environmental and functional constraints, and general operating conditions shall comply with the tool's evaluation criteria and qualification. {ISO 26262-8, Section 11.4.3.1}
9. If the tool confidence level evaluation or qualification of a software tool is performed independently from the development of a particular safety-related item or element, the validity of this predetermined tool confidence level or qualification shall be confirmed prior to the software tool being used for the development of a particular safety-related item or element. {ISO 26262-8, Section 11.4.2.1}

5.1.1.7 *Software Tool Training*

1. Procedures should be developed to ensure that all persons involved in any overall system or software life cycle activity, including activities for verification, management of functional safety, and functional safety assessment shall have the appropriate competence including training, technical knowledge, experience, and qualifications relevant to the specific duties

that they have to perform and such procedures shall include requirements for the refreshing, updating, and continued assessment of competence. {IEC 61508-1, Section 6.2.13}

5.1.1.8 Software Tool Configuration Management

1. Where automatic or semi-automatic tools are used for the production of documentation, specific procedures may be necessary to ensure effective measures are in place for the management of versions or other control aspects of the documents. {IEC 61508-1, Section 5.2.11}
2. Software configuration management (SCM) shall maintain accurately and with unique identification all configuration items which are necessary to meet the safety integrity requirements of the safety-related system including all tools and development environments which are used to create or test, or carry out any action on, the software of the safety-related system. {IEC 61508-3, Section 6.2.3.c}
3. The CM system shall cover software development and the software development environment necessary for the reproducibility of the development and for the maintenance activities including all the tools, translators, data and test files, parameterization files, and supporting hardware platforms. {BS EN 50128, Section 6.5.4.12}
4. Where off-line support tools of classes T2 and T3 generate items in the configuration baseline, CM shall ensure that information on the tools is recorded in the configuration baseline. This includes the identification of the tool and its version; the identification of the configuration baseline items for which the tool version has been used; and the way the tool was used (including the tool parameters, options, and scripts selected) for each configuration baseline item so the baseline can be reconstructed. {IEC 61508-3, Section 7.4.4.15}
5. Configuration management shall ensure that for tools in classes T2 and T3, only qualified versions are used. {IEC 61508-3, Section 7.4.4.16; BS EN 50128, Section 6.7.4.10}
6. Configuration management shall ensure that only tools compatible with each other and with the safety-related system are used recognizing that the safety-related system hardware may also impose compatibility constraints on software tools (e.g., a processor emulator needs to be an accurate model of the real processor electronics). {IEC 61508-3, Section 7.4.4.17}

5.1.1.9 Software Tool Review and Approval

1. Review and approval of software tools is dependent on following all the objectives, executing all activities that satisfy each objective, and developing all the associated life cycle data required by RTCA DO-178C, RTCA DO-330, and all its supplements. {FAA AC 20-115C, Section 6.a and 6.b}

5.1.2 Baseline Guidance Additions

5.1.2.1 *Software and Software Tool Classifications*

1. Software off-line support tools may be divided into the following classes: (IEC 61508-4, Section 3.2.11; BS EN 50128, Sections 3.1.42 through 3.1.44)
 - a. T1: Generates no outputs which can directly or indirectly contribute to the executable code (including data) of the safety-related system (e.g., text editor, requirements or design support tool with no automatic code generation capabilities, and configuration control tools);
 - b. T2: Supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software (e.g., test harness generator, test coverage measurement tool, and static analysis tool);
 - c. T3: Generates outputs which can directly or indirectly contribute to the executable code of the safety-related system (e.g., optimizing compiler where the relationship between the source code program and the generated object code is not obvious and a compiler that incorporates an executable run-time package into the executable code).
2. If the correct selection of TI or TD is unclear or doubtful, TI and TD should be estimated conservatively. {ISO 26262-8, Section 11.4.5.3}
3. A tool detection error class of TD3 typically applies if there are no systematic measures in the development process available, and therefore, malfunctions of the software tool and their corresponding erroneous outputs can only be detected randomly. {ISO 26262-8, Section 11.4.5.2}
4. A tool detection error class of TD1 can be chosen for a code generator when the produced source code is verified. {ISO 26262-8, Section 11.4.5.2}

5.1.2.2 *Software Tool Planning*

No new software tool planning guidance has been added.

5.1.2.3 *Software Tool Documentation*

1. The documentation of the results of tool validation should cover the following results: {IEC 61508-3, Section 7.4.4.7; BS EN 50128, Section 6.7.4.5}
 - a. a chronological record of the validation activities,
 - b. the version of the tool product manual being used,
 - c. the tool functions being validated,

- d. tools and equipment used,
 - e. the results of the validation activity including a statement that the software has passed the validation or the reason for its failure,
 - f. test cases and their results for subsequent analysis, and
 - g. discrepancies between expected and actual results.
2. Tool settings used during the development, verification, or validation of baseline equipment, software, or HDL configured devices should be recorded in the development records. This is useful not only for the final software consistency; it also helps in assessing the origin of a fault, which might lie in the source code, in the tool, or in the tool settings. Information about the tool settings used may be critical to assessing the potential for common cause failures (CCFs) due to software tools. {IAEA DS-431, Sections 7.163 and 7.164}
 3. A software requirements specification should allow for the capabilities of the computers, tools, and similar existing systems to ensure that the software requirements are feasible. {IAEA DS-431, Section 9.11.h}
 4. The software design documentation should include those implementation constraints that need to be observed during the design phase. Such implementation constraints may include any need to ensure diversity, and particular attributes of the programming language, compilers, subroutine libraries, and any other supporting tools. These constraints should be justified or be traceable to higher-level requirements or constraints. {IAEA DS-431, Sections 9.37 through 9.39}
 5. A software verification plan should be produced that documents the functionality of any verification tool, including expectations and limitations on how it is to be used (e.g., domain, language, process). {IAEA DS-431, Section 9.66.h}

5.1.2.4 Software Tool Selection and Use

1. Corresponding tools should be selected for each sub-phase of software development including guidelines for their application. {ISO 26262-6, Section 5.4.5}
2. Success in achieving systematic safety integrity depends on selecting techniques with attention to the consistency and the complementary nature of the chosen methods, languages, and tools for the whole development cycle; whether the developers use methods, languages, and tools they fully understand; and whether the methods, languages, and tools are well-adapted to the specific problems encountered during development. {IEC 61508-3, Section 7.1.2.7}
3. Appropriate off-line tools to support the development of software should be used in order to increase the integrity of the software by reducing the likelihood of introducing or not detecting faults during the development. Examples of tools relevant to the phases of the

software development life cycle include: {IEC 61508-3, Section 7.4.4.2; BS EN 50128, Section 6.7.4.1}

- a. transformation or translation tools that convert a software or design representation (e.g., text or a diagram) from one abstraction level to another level: design refinement tools, compilers, assemblers, linkers, binders, loaders, and code generation tools;
 - b. V&V tools such as static code analyzers, test coverage monitors, theorem proving assistants, and simulators;
 - c. diagnostic tools used to maintain and monitor the software under operating conditions;
 - d. infrastructure tools such as development support systems;
 - e. configuration control tools such as version control tools; and
 - f. application data tools that produce or maintain data which are required to define parameters and to instantiate system functions including function parameters, instrument ranges, alarm and trip levels, output states to be adopted at failure, and geographical layout.
4. Off-line support tools should be selected to be integrated so they work co-operatively such that the outputs from one tool have suitable content and format for automatic input to a subsequent tool, thus minimizing the possibility of introducing human error in the reworking of intermediate results. {IEC 61508-3, Section 7.4.4.2}
 5. Off-line support tools should be selected to be compatible with the needs of the application, of the safety-related system, and of the integrated toolset. {IEC 61508-3, Section 7.4.4.2}
 6. The availability of suitable tools to supply the services that are necessary over the whole lifetime of the safety-related system should be considered. {IEC 61508-3, Section 7.4.4.2}
 7. Consideration should be given to the competence of the users of the selected tools. {IEC 61508-3, Section 7.4.4.2}
 8. Tools should be used to support all aspects of the I&C development life cycle where benefits result through their use and where tools are available. {IAEA DS-431, Section 7.148}
 9. The benefits and risk of using a tool should be balanced against the benefits and risk of not using a tool. The important principle is to choose tools that limit the opportunity for making errors and introducing faults, but maximize the opportunity for avoiding or detecting faults. {IAEA DS-431, Sections 7.152 and 7.153}
-

-
10. Tools should be selected to remain available throughout the system's service life and be compatible with other tools used during system development. {IAEA DS-431, Section 7.154}
 11. The functionality and limits of applicability of all tools should be identified and documented. {IAEA DS-431, Section 7.155}
 12. The tools and their output should not be used outside their declared functionality or limits of application without prior justification. For example, tools cannot replace humans when judgment is involved. In some cases, tool support is more appropriate than complete automation of a process. {IAEA DS-431, Sections 7.156 and 7.157}
 13. A suitable set of implementation tools should be selected with the aim of minimizing error. {IAEA DS-431, Section 9.58}

5.1.2.5 *Software Tool Development, Qualification, and Dedication*

1. The qualification of each new version of an off-line support tool may rely on evidence provided for an earlier version if sufficient evidence is provided that: {IEC 61508-3, Section 7.4.4.18; BS EN 50128, Section 6.7.4.11}
 - a. the functional differences (if any) will not affect tool compatibility with the rest of the toolset; and
 - b. the new version is unlikely to contain significant new, unknown faults.
 - i. Evidence that the new version is unlikely to contain significant new, unknown faults may be based on:
 - (a) a clear identification of the changes made,
 - (b) an analysis of the verification and validation (V&V) actions performed on the new version, and
 - (c) any existing operational experience from other users that is relevant to the new version.
2. The fitness for purpose of a non-trusted compiler can be justified by subjecting the object code produced by the compiler to a combination of tests, checks, and analyses that are capable of ensuring the correctness of the code to the extent consistent with the target SIL and in particular: {BS EN 50128, Section 6.7.4.6}
 - a. testing has sufficiently high coverage of the implemented code and any unreachable code by testing has been shown by checks or analyses that the function concerned is executed correctly;

-
- b. checks and analyses have been applied to the object code and shown to be capable of detecting the types of errors which might result from a defect in the compiler;
 - c. no more translation with the compiler has taken place after testing, checking, and analysis; and
 - d. if further compilation or translation is carried out, all tests, checks, and analyses will be repeated.
 3. The assessment of a translator may be performed for a specific application project, or for a class of applications. In the latter case all necessary information on the tool (specification or product manual) regarding the intended and appropriate use of the tool should be available to the user of the tool. The assessment of the tool for a specific project may then be reduced to checking general suitability of the tool for the project and compliance with the specification or product manual (i.e., proper use of the tool). Proper use might include additional verification activities within the specific project. {IEC 61508-3, Section 7.4.4.10}
 4. A validation suite (i.e., a set of test programs whose correct translation is known in advance) may be used to evaluate the fitness for purpose of a translator according to defined criteria, which should include functional and non-functional requirements. For the functional translator requirements, dynamic testing may be a main validation technique. If possible, an automatic testing suite should be used. {IEC 61508-3, Section 7.4.4.10}
 5. Prevention or detection of software tool erroneous output can be accomplished through process steps, redundancy in tasks, redundancy in software tools, or by rationality checks within the software tool itself. {ISO 26262-8, Section 11.4.5.2}
 6. Tools should be verified and assessed consistent with the tool reliability requirements, the type of tool, the potential of the tool to introduce faults or fail to make the user aware of existing faults, and the extent to which the tool may affect redundant elements of a system or diverse systems. Examples of situations that can affect the degree of verification and assessment needed include, for example: {IAEA DS-431, Sections 7.158 and 7.159}
 - a. tools that have the ability to introduce faults need to be verified to a greater degree than tools that are demonstrated to not have that capability,
 - b. tools that can fail to make the user aware of existing faults need to be verified to a greater degree than tools that do not have that capability,
 - c. verification is not necessary for tools when the output of the tool is systematically and independently verified,
 - d. less rigor in tool verification may be accepted if there is mitigation of any potential tool faults (e.g. by process diversity or system design).
-

-
7. The verification and assessment of software tools should take into account experience from prior use, including experience of the developers and experience gained from the processes in which the tools are used. {IAEA DS-431, Section 7.160}

5.1.2.6 Software Tool Quality Assurance

1. For each tool in class T3, evidence that the output of the tool conforms to its specification or that documentation of the output or failures in the output are detected may be based on: {IEC 61508-3, Section 7.4.4.6; BS EN 50128, Section 6.7.4.4}
 - a. a suitable combination of history of successful use in similar environments and for similar applications (within the organization or other organizations),
 - b. tool validation,
 - c. diverse redundant code which allows the detection and control of failures resulting in faults introduced by a tool, and
 - d. other appropriate methods for avoiding or handling failures introduced by tools such as version history or a record of the errors and ambiguities associated with tool use in the environment.
2. Software tool usage, environmental and functional constraints, and general operating conditions can be shown to comply with the tool's evaluation criteria or its qualification through the use of identical version and configuration settings for the same use cases together with the same implemented measures for the prevention or detection of malfunctions and their corresponding erroneous output, as documented in the qualification report for the software tool. {ISO 26262-8, Section 11.4.3.1}
3. The choice, verification, and assessment of tools should be justified and documented. {IAEA DS-431, Section 7.161}
4. The software implementation should be established using a predetermined combination of techniques commensurate with the system's importance to safety, covering languages, tools, coding practices, analysis, review, and testing. {IAEA DS-431, Section 9.43.b}

5.1.2.7 Software Tool Training

No new software tool training guidance has been added.

5.1.2.8 Software Tool Configuration Management

1. A version history may provide assurance of maturity of the tool, and a record of the errors and ambiguities that should be taken into account when the tool is used in the new development environment. {IEC 61508-3, Section 7.4.4.6}
2. All tools should be under appropriate CM. {IAEA DS-431, Section 7.162}

5.1.2.9 **Software Tool Review and Approval**

1. A liaison process should be used as part of the software development life cycle process to establish communication and understanding between the applicant and the approval authority. The approval authority may review the software life cycle processes and data to assess compliance to applicable requirements. {FAA Order 8110.49, Section 2.1.a}
2. The amount of approval authority involvement in a software project should be determined as HIGH, MEDIUM, or LOW based on the software level of the project, the amount and quality of review support, the experience and history of the applicant and software developer, service difficulty history, and several other factors. {FAA Order 8110.49, Section 3-2}
3. The use of unusual tools will increase the level of approval authority involvement. {FAA Order 8110.49, Section 3-2.c}
 - a. For software level A or B projects, the use of unusual tools requires at least a HIGH level of approval authority involvement.
 - b. For software level C or D projects, the use of unusual tools requires at least a MEDIUM level of approval authority involvement.

5.2 **Baseline Deletions**

As part of the baseline analysis and using the results of the document analysis from Section 2, baselines have been deleted where requirements, guidance, or review and approval practices are not suitable for NRC use, or are redundant with other guidance or review and approval practices. Deleted baseline requirements are identified in Section 5.2.1 and deleted guidance, review, and approval practices are identified in Section 5.2.2. Reasons have been given for every deleted baseline.

5.2.1 **Baseline Requirement Deletions**

1. If it cannot be demonstrated that defects not detected by software tools or introduced by software tools will be detected by V&V activities, the software tool should be designed as safety-related software itself, with all the attendant regulatory requirements for safety software. {NUREG 0800 Appendix 7.1-D, Clause 5.3.2; IEEE 7-4.3.2-2003, Clause 5.3.2.b}

Reason: This requirement is considered unacceptable by the NRC, extremely difficult if not impossible to satisfy, and has been superseded by more recent requirements in IEEE 7-4.3.2-2010.
2. The equipment families used for the development of an I&C system should be associated with software tools that can reduce the risk of introducing faults in the configuration of their pre-developed software and in the design of the system. {IEC 62138 Ed. 1.0B, Clause 5.1.4}

Reason: This requirement was consolidated with a similar requirement.

3. The software tools that might lead to faults already existing in software or in system design being overlooked should be selected and used in a way which reduces this risk. Their use should be traced. {IEC 62138 Ed. 1.0B, Clause 6.1.4}

Reason: This requirement was consolidated with a similar requirement.

4. Libraries which are used in the target system shall be considered as sets of pre-developed software components. Those components of the library used shall be evaluated, qualified, and used in accordance with the requirements of pre-developed software. {IEC 60880 Ed.2, Clause 14.3.4.5}

Reason: This requirement applies to pre-developed software and not to software tools.

5. Integrity-level determination quantifies the complexity, criticality, risk, safety level, security level, desired performance, reliability, or other system-unique characteristics based on the importance of the system to the user and acquirer. The degree of rigor and intensity in performing and documenting the task shall be commensurate with the integrity level. {IEEE 1012-2012, Clause 5}

Reason: This requirement applies to integrity levels of software and does not apply to software tools.

6. The integrity level assigned to reused, COTS, and government off-the-shelf (GOTS) tools shall be in accordance with the integrity-level scheme adopted for the system element in which COTS or GOTS tools are used for the project. The reused COTS or GOTS tools shall be evaluated for use in the context of its application. The design, development, procedural, and technology features implemented in the system can raise or lower the assigned integrity levels. {IEEE 1012-2012, Clause 5}

Reason: This requirement applies to reused, COTS, and GOTS software and does not apply to software tools.

7. If V&V of reused software tools cannot be accomplished at the appropriate level, then the items may be used so long as the risk associated with this use is recognized and accounted for in the risk mitigation strategy. The V&V effort should assure that the risks are thoroughly understood, properly documented, and properly tracked under the risk analysis tasks. {IEEE 1012-2012, Annex D.3}

Reason: This requirement applies to reused, COTS, and GOTS software and does not apply to software tools.

5.2.2 Baseline Guidance Deletions

1. A software tool may be qualified only for use on a specific system. Tool qualification data must be generated, a tool qualification plan must be generated and followed, and tool

operational requirements must be specified to describe the tool's operational functionality including all functions and technical features. {RTCA DO-178B, Section 12.2}

Reason: Guidance in RTCA DO-178B has been superseded by guidance in RTCA DO-178C.

2. If a life cycle process is eliminated, reduced, or automated by the use of software development tools and the tool output has not been verified, then the software development tools need to be treated at a level equivalent to the safety-related plant software and that the tool meets its tool-specific operational requirements. A trial period may be needed during which a verification of tool output is performed and tool-related problems are analyzed, recorded, and corrected to demonstrate that the tool meets its operational requirements. {RTCA DO-178B, Section 12.2.1.c}

Reason: Guidance in RTCA DO-178B has been superseded by guidance in RTCA DO-178C.

3. The tool qualification plan should explain the methods that will be used to qualify the tool, including those objectives and guidance that will be met using a service history approach. {RTCA DO-248B, Section 12.3.4}

Reason: Guidance in RTCA DO-248B has been superseded by guidance in RTCA DO-248C.

4. If the compiler used to generate executable code uses optimization, the correctness of the optimization need not be verified if the test cases give coverage consistent with the software level. Otherwise, the impact of optimization should be determined. {RTCA DO-178B and RTCA DO-178C, Section 4.4.2.a}

Reason: This guidance is redundant with other baseline guidance on the treatment of compilers.

5. Compilers may also produce object code that is not directly traceable to source code, for example, initialization, built-in error detection, or exception handling. The software planning process should provide a means to detect this object code and to ensure verification coverage. {RTCA DO-178B and RTCA DO-178C, Section 4.4.2.b}

Reason: This guidance is redundant with another baseline guidance on the treatment of compilers.

6. Reviewers should thoroughly evaluate tool usage. {NUREG-0800 Appendix 7.1-D, Section 5.3.2}

Reason: This guidance is redundant with another baseline guidance.

7. Software verification tools can be qualified by just confirming that the tool meets its tool-specific operational requirements under normal operational conditions. {RTCA DO-178B, Section 12.2.2}

Reason: This guidance has been superseded by guidance in RTCA DO-178C.

8. The functional requirements of a verification tool should accurately reflect the corresponding life cycle process objective that it is replacing or simplifying. When selecting a tool, the first process should be how well the required life cycle process objectives can be satisfied by the tool. The objectives must be readily understood. If they are not, it is not possible to correctly determine that the tool has implemented the objectives correctly. {RTCA DO-248B, Section 4.1.2.1}

Reason: This guidance has been superseded by tool selection guidance in RTCA DO-178C and RTCA DO-248C.

9. The interpretation of a tool's output is based upon the user's understanding of the tool's function. If an incorrect assumption has been made about the tool's function, the interpretation of its output may be equally flawed. When selecting a tool it is essential to understand its limitations so that additional manual efforts may augment the tool to complete the verification process. {RTCA DO-248B, Sections 4.1.2.3.1 and 4.1.2.3.2}

Reason: This guidance has been superseded by tool selection guidance in RTCA DO-178C and RTCA DO-248C.

10. If a single software tool is used during several phases of the software development life cycle or a single software tool is used in the same development life cycle for software in multiple systems, the tool must be sufficiently verified and validated to ensure that a defect in the tool does not cause a fault that can result in a common-cause failure in the software. {IEEE 603-1991 and IEEE 603-2009}

Reason: This guidance is not specifically discussed in the IEEE standards and represented an ISL observation.

5.3 Final Baselines

This section contains final baseline requirements and baseline guidance, review, and approval practices. The baselines are organized into nine categories related to tool specific activities. This organization of the baselines provides a better understanding of the type and applicability of the tool requirements, guidance, and review and approval practice. Within each of the nine tool specific activity categories, the final baseline requirement is listed first, followed by all baseline guidance applicable to that requirement.

5.3.1 Software and Software Tool Classifications

Requirement 5.3.1.1: Functions to be performed by I&C systems shall be assigned to categories according to their importance to safety that is identified by means of the consequences in the event of its failure when it is required to be performed and the consequences in the event of a spurious actuation. {IEC 61226, Section 5.1}

Guidance 5.3.1.1.1: COTS software, including software tools such as compilers, linkers, automatic configuration managers, or the like, should be classified and designed as IEC

61226 safety class A, B, C, or unclassified depending on the level of software produced and whether there exists a diverse alternative or another software tool that verifies the output. The following criteria should be used to determine the safety category classification of COTS software. {NUREG/CR-6421, Tables 2 and 3}

- a. Criterion 1 - If the COTS product is used directly in a system important to safety, the COTS safety category is determined by the criteria of IEC 61226.
- b. Criterion 2 - If COTS that directly produces or controls the configuration of an executable software product is used in a system important to safety, and if no method exists to validate the output of the COTS product, then the COTS safety category is the same as that of its output, except that Category C software may be produced by COTS products of the unclassified category. COTS software that directly produces Category A or B software that is validated by other means is Category B or C, respectively.
- c. Criterion 3 - COTS that supports production of Category A, B, or C software, but does not directly produce or control the configuration of such software modules, falls under the safety category "unclassified."
- d. Criterion 4 - If the COTS product has no impact on category A, B, or C software or systems, it falls under the safety category "unclassified".

Requirement 5.3.1.2: A software on-line support tool shall be considered to be a software element of the safety-related system. {IEC 61508-3, Section 7.4.4.1}

Requirement 5.3.1.3: Software off-line support tools are those that support a phase of the software development life cycle and that cannot directly influence the safety-related system during its run time. (IEC 61508-4, Section 3.2.11; BS EN 50128, Sections 3.1.42 through 3.1.44)

Guidance 5.3.1.3.1: Software off-line support tools may be divided into the following classes: (IEC 61508-4, Section 3.2.11; BS EN 50128, Sections 3.1.42 through 3.1.44)

- a. T1: Generates no outputs which can directly or indirectly contribute to the executable code (including data) of the safety-related system (e.g., text editor, requirements or design support tool with no automatic code generation capabilities, and configuration control tools);
- b. T2: Supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software (e.g., test harness generator, test coverage measurement tool, and static analysis tool);
- c. T3: Generates outputs which can directly or indirectly contribute to the executable code of the safety-related system (e.g., optimizing compiler where the relationship between the source code program and the generated object

code is not obvious and a compiler that incorporates an executable run-time package into the executable code).

Requirement 5.3.1.4: The intended usage of the software tool shall be analyzed and evaluated to determine the tool impact (TI) class and the tool error detection (TD) class. The TI class and the TD class are used to determine the tool confidence level (TCL). {ISO 26262-8, Sections 11.4.5.2 through 11.4.5.5}

- a. The TI class is a measure of the possibility that a malfunction of a particular software tool can introduce or fail to detect errors in a safety-related item or element being developed.
 - i. Class TI1 shall be selected when there is an argument that there is no such possibility.
 - ii. Class TI2 shall be selected in all other cases.
- b. The TD class is an expression of the confidence in measures that prevent the software tool from malfunctioning and producing corresponding erroneous output, or in measures that detect that the software tool has malfunctioned and has produced corresponding erroneous output. If a software tool is used for the tailoring of the development process in such a way that activities or tasks are omitted, TD2 shall not be selected.
 - i. Class TD1 shall be selected if there is a high degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected.
 - ii. Class TD2 shall be selected if there is a medium degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected.
 - iii. Class TD3 shall be selected in all other cases
- c. The required software TCL shall be determined according to Table 5.4 based on the TI and TD classes.

Table 5.4 – Determination of the Tool Confidence Level (TCL)

		Tool error detection		
		TD1	TD2	TD3
Tool impact	TI1	TCL1	TCL1	TCL1
	TI2	TCL1	TCL2	TCL3

Guidance 5.3.1.4.1: If the correct selection of TI or TD is unclear or doubtful, TI and TD should be estimated conservatively. {ISO 26262-8, Section 11.4.5.3}

Guidance 5.3.1.4.2: A tool detection error class of TD3 typically applies if there are no systematic measures in the development process available, and therefore, malfunctions of the software tool and their corresponding erroneous outputs can only be detected randomly. {ISO 26262-8, Section 11.4.5.2}

Guidance 5.3.1.4.3: A tool detection error class of TD1 can be chosen for a code generator when the produced source code is verified. {ISO 26262-8, Section 11.4.5.2}

Requirement 5.3.1.5: If the tool eliminates, reduces, or automates software life cycle processes and its output is not verified, the tool must be qualified and the following procedure should be used to determine the tool qualification level (TQL). {RTCA DO-178C, Sections 2.3.2, 2.3.3, 12.2.2, and 12.2.3}

- a. Each tool should be assigned a criterion as determined below based on the impact of the tool use on the software life cycle processes.

Criterion 1: A tool whose output is part of the airborne software and thus could insert an error.

Criterion 2: A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of:

1. Verification process(es) other than that automated by the tool, or
2. Development process(es) that could have an impact on the airborne software.

Criterion 3: A tool, which within the scope of its intended use, could fail to detect an error.

- b. Each tool should be assigned a software level as determined below consistent with the software component developed with each tool based upon the failure condition which may result from anomalous behavior of the software as determined by the system safety assessment.

Level A: Software whose anomalous behavior would cause or contribute to a failure of system function resulting in a catastrophic failure condition for the aircraft which would result in multiple fatalities, usually with the loss of the airplane.

Level B: Software whose anomalous behavior would cause or contribute to a failure of system function resulting in a hazardous failure condition for the aircraft which would reduce the capability of the airplane or the ability of the flight crew to cope with adverse operating conditions to the extent that there would be:

1. a large reduction in safety margins or functional capabilities,

2. physical distress or excessive workload such that the flight crew cannot be relied upon to perform their tasks accurately or completely, or
3. serious or fatal injury to a relatively small number of the occupants other than the flight crew.

Level C: Software whose anomalous behavior would cause or contribute to a failure of system function resulting in a major failure condition for the aircraft which would reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions to the extent that there would be:

1. a significant reduction in safety margins or functional capabilities,
2. a significant increase in crew workload or in conditions impairing crew efficiency, or discomfort to the flight crew, or
3. physical distress to passengers or cabin crew, possibly including injuries.

Level D: Software whose anomalous behavior would cause or contribute to a failure of system function resulting in a minor failure condition for the aircraft which would not significantly reduce airplane safety, and which involve crew actions that are well within their capabilities including:

1. a slight reduction in safety margins or functional capabilities,
2. a slight increase in crew workload, such as routine flight plan changes, or
3. some physical discomfort to passengers or cabin crew.

Level E: Software whose anomalous behavior would cause or contribute to a failure of system function with no effect on aircraft operational capability, safety, or crew workload. If a software component is determined to be Level E and this is confirmed by the certification authority, no tool qualification is required.

- c. Each tool should be assigned an appropriate TQL as determined from Table 5.5 based on the assigned tool criterion and the software level.

Table 5.5 – Tool Qualification Level Determination

Software Level	Tool Criterion		
	1	2	3
A	TQL-1	TQL-4	TQL-5
B	TQL-2	TQL-4	TQL-5
C	TQL-3	TQL-5	TQL-5
D	TQL-4	TQL-5	TQL-5
E	No tool qualification necessary		

5.3.2 Software Tool Planning

Requirement 5.3.2.1: All project tools that could potentially impact safety-critical software, the degree of impact, and mitigation strategies shall be discussed in the project plan. {NASA-STD-8719.13B, Clause 5.11.1.1}

Requirement 5.3.2.2: Installation of upgrades to approved tools, withdrawal of previously approved tools, and identification of limitations of the tools should be identified in the project plan. {NASA-STD-8719.13B, Clause 5.11.1.2}

Requirement 5.3.2.3: The process and criteria used to select, approve, and control the tools shall be described in the project plan. {NASA-STD-8719.13B, Clause 5.11.1.2}

Requirement 5.3.2.4: Tools to be used should be identified in the development plan. {IAEA NS-G-1.1, Section 4.6}

Guidance 5.3.2.4.1: The software development plan should include the following resource characteristics. {NUREG-0800 BTP 7-14, Section B.3.1.2.3; RTCA DO-330, Section 4.1}

- a. a description of the software development methods, techniques, and tools to be used;
- b. a description of the approach to be followed for reusing software;
- c. identify suitable facilities, tools, and aids to facilitate the production, management, and publication of appropriate and consistent documentation and for the development of the software;
- d. describe the software development environment, including software design aids, compilers, loaders, and subroutine libraries;
- e. require that tools be qualified with a degree of rigor and level of detail appropriate to the safety significance of the software that is to be developed using the tools;
- f. the tool qualification level is defined; and
- g. the tool operational environment is described.

Requirement 5.3.2.5: The entire project should be planned in advance to select an integrated set of tools. {IAEA NS-G-1.1, Section 4.6}

Guidance 5.3.2.5.1: The software planning process should meet the following objectives. {RTCA DO-178C, Clause 4.1}

- a. The software life cycle environment, including the methods and tools to be used for the activities of each software life cycle process, has been selected and defined.
- b. Software plans should define the transition criteria by specifying availability of tools, methods, plans, and procedures.

Guidance 5.3.2.5.2: Activities for the software planning process include: {RTCA DO-178C, Clause 4.2}

- a. Methods and tools should be chosen that aid error prevention and provide defect detection in the software development process.
- b. Methods and tools should be chosen to achieve dissimilarity necessary to satisfy the system safety objectives when multiple-version dissimilar software is used in a system to mitigate software failures.
- c. If user-modifiable software is planned, related processes, tools, environment, and data items substantiating the design should be specified in the software plans and standards.
- d. When parameter data items are planned, the processes to develop, verify, and modify parameter data items, and tool qualification should be addressed.

Guidance 5.3.2.5.3: The implementation of certain compiler features may produce object code that is not directly traceable to the source code, for example, initialization, built-in error detection, or exception handling. The software planning process should provide a means to detect this object code and to ensure verification coverage, and should define the means in the appropriate plan. {RTCA DO-178C, Clause 4.4.2}

Requirement 5.3.2.6: A software QA plan should be produced at the outset of the project including identification of tools to be used on the project and the procedures for qualifying or dedicating the tools. {IAEA NS-G-1.1, Section 4.11}

Requirement 5.3.2.7: The QA plan should require that tools used are identified, documented, known to, and mastered by, the persons concerned. {IEC 60880 Ed. 2.0, Clause 5.5.8}

Requirement 5.3.2.8: Methods, measures, and tools for QA according to the allocated SILs shall be specified or referenced in the software QA plan. {BS EN 50128, Section 6.5.4.5.e}

Requirement 5.3.2.9: The QA plan shall distinguish the tools that might introduce faults in software or in system design, from those which only might lead to overlooking already existing faults. {IEC 62138 Ed. 1.0B, Clause 6.1.4}

Requirement 5.3.2.10: An overall safety validation plan shall be developed that shall include the required environment in which the validation activities are to take place (e.g., for tests this would include calibrated tools and equipment). {IEC 61508-1, Section 7.8.2.1.h}

Requirement 5.3.2.11: Planning for the validation of a safety-related system shall consider the required environment in which the testing is to take place including all necessary tools and equipment (also plan which tools and equipment should be calibrated). {IEC 61508-2, Section 7.3.2.2.d}

Requirement 5.3.2.12: The validation plan for software aspects of system safety shall consider the required environment in which the validation activities are to take place (for example, for tests this could include calibrated tools and equipment). {IEC 61508-3, Section 7.3.2.2.g}

Requirement 5.3.2.13: Safety-related system verification planning shall be carried out for each phase of the safety life cycle and address all the criteria, techniques, and tools to be utilized in the verification for that phase. {IEC 61508-2, Section 7.9.2.2; ISO 26262-8, Section 9.4.1.1.e}

Guidance 5.3.2.13.1: If a new compiler, linkage editor, or loader version is introduced, or compiler options are changed during the software life cycle, previous tests and coverage analyses may no longer be valid. The verification planning should provide a means of re-verification. {RTCA DO-178C, Clause 4.4.2}

Requirement 5.3.2.14: The software verification planning shall refer to the criteria, techniques, and tools to be used in the verification activities, and shall address the selection and utilization of verification tools (test harness, special test software, input/output simulators, etc.) {IEC 61508-3, Section 7.9.2.2.c}

Requirement 5.3.2.15: The usage of a software tool shall be planned, including the determination of: {ISO 26262-8, Section 11.4.4.1}

- a. the identification and version number of the software tool;
- b. the configuration of the software tool;
- c. the use cases of the software tool;
- d. the environment in which the software tool is executed;
- e. the maximum SIL of all the safety requirements, allocated to the item or element that can be violated, if the software tool is malfunctioning and producing corresponding erroneous output; and
- f. the methods to qualify the software tool, if required based on the determined level of confidence.

Requirement 5.3.2.16: If qualified tools are used to develop or verify a reusable software component, the reusable software component developer must consider reuse of those tools during reusable software component development and acceptance. {FAA AC 20-148, Section 11.f}

- a. For development tools, the tool qualification plan and the tool accomplishment summary must document the portions of the tool qualification that the applicant is required to complete.
- b. For verification tools, the plan for software aspects of certification and the software accomplishment summary must document the portions of the tool qualification that the applicant is required to complete.
- c. The reusable software component developer must provide tool plans, tool operational requirements, and the tool accomplishment summary to the applicant for all tools used in getting acceptance of the reusable software component.

Guidance 5.3.2.16.1: Any assumptions and limitations that affect proper reusable tool operation should also be identified in the project plan (as well as in the tool user's guide). {Position Paper CAST-22, Section 4.5}

5.3.3 Software Tool Documentation

Requirement 5.3.3.1: All software tools shall be identified and shall be addressed within required software development activities and documentation. {FAA-STD-026A, Clause 4.4}

Guidance 5.3.3.1.1: The functionality and limits of applicability of all tools should be identified and documented. {MDEP DICWG-02, Position 3}

Guidance 5.3.3.1.2: Documentation associated with computer-aided software engineering (CASE) tools, data in these tools, compilers, test tool, test data, simulations, emulations, utilities, CM tools, databases and data files, and other software shall include: {FAA-STD-026A, DID-FAA-026-03, Clause 10.2.3.3}

- a. specific names, identification numbers, version numbers, release numbers, and configurations;
- b. rationale for the selected software;
- c. reference to user/operator manuals or instructions for each item, as applicable;
- d. identification of each software item and document as Government-furnished, as item that will be delivered to the support organization, as item the support organization is known to have, as item the support organization must acquire, or other description of status;

- e. if items must be acquired, information about a current source of supply, including whether the item is currently available and whether it is expected to be available at the time of delivery;
- f. information about vendor support, licensing, and data rights, including whether the item is currently supported by the vendor, whether it is expected to be supported at the time of delivery, whether licenses will be assigned to the support organization, and the terms of such licenses; and
- g. security and privacy considerations, limitations, or other items of interest.

Requirement 5.3.3.2: When activities of the software development life cycle process are automated using software tools, the activities automated shall be documented including the documentation of the inputs and outputs relevant to the life cycle phase. {IEC 60880 Ed.2, Clause 5.4.6}

Guidance 5.3.3.2.1: If requirements are generated using specification or representation tools, traceability should be maintained between these representations and the natural language descriptions of the software requirements that are derived from systems requirements and system safety analyses. {RG 1.172, Section C.2.a}

Guidance 5.3.3.2.2: The software requirements specification should allow for the capabilities of the computers, tools, and similar existing systems to ensure that the software requirements are feasible. {IAEA DS-431, Section 9.11.h}

Guidance 5.3.3.2.3: The software design documentation should include those implementation constraints that need to be observed during the design phase. Such implementation constraints may include any need to ensure diversity, and particular attributes of the programming language, compilers, subroutine libraries, and any other supporting tools. These constraints should be justified or be traceable to higher-level requirements or constraints. {IAEA DS-431, Sections 9.37 through 9.39}

Requirement 5.3.3.3: It shall be possible to identify all translation tools and tool versions used to produce each executable entity. {IEC 60880 Ed.2, Clause 5.6.12}

Requirement 5.3.3.4: Any separately compiled modules that are included by an automatic code generator should be supported by separate documents for the software requirements. {IAEA NS-G-1.1, Section 7.4}

Requirement 5.3.3.5: Automated validation tools that generate test data, transport, or transform test data and test results, and evaluate test results should record a complete test log. This is applicable to module tests as well as to plant simulations. {IEC 60880 Ed.2, Clause 14.3.6.1}

Requirement 5.3.3.6: Software tool use shall be traced so that the tools, if any, which were used to generate a given item or information may be identified. {IEC 62138 Ed. 1.0B, Clause 6.1.4}

Requirement 5.3.3.7: Software testing shall be documented by a test specification that shall document the test environment, tools, configuration, and programs used for testing. {BS EN 50128, Section 6.1.4.4.d}

Guidance 5.3.3.7.1: Repetition of test documentation is sometimes needed when automated tools are used for testing. Test information must be made easily accessible to improve the timeliness of a safety conclusion. {RG 1.170, Section C.8}

Guidance 5.3.3.7.2: Documentation used to support software testing must include as a minimum environmental conditions and special controls, equipment, tools, and instrumentation needed for the accomplishment of testing. {RG 1.170, Section C.1.c}

Requirement 5.3.3.8: The information documented during the overall safety validation process shall include tools and equipment used, along with calibration data. {IEC 61508-1, Section 7.14.2.3}

Requirement 5.3.3.9: The verification plan shall document or refer to the criteria, techniques, and tools to be used in the verification activities. {IEC 61508-1, Section 7.18.2.2; BS EN 50128, Section 6.2.4.5}

Guidance 5.3.3.9.1: The information needed to reach a determination that the software tools are adequate for their intended use should be contained in the documentation of the software tool verification program. The intended use of those tools should be described in the software development plan, software integration plan, and software test plan, depending on how the tool will be used. Use of the tools may be audited to assess the quality of the tools. {DI&C-ISG-06, Section D.10.4.2.3.2}

Guidance 5.3.3.9.2: A software verification plan should be produced that documents the functionality of any verification tool, including expectations and limitations on how it is to be used (e.g., domain, language, process). {IAEA DS-431, Section 9.66.h}

Requirement 5.3.3.10: The output of the functional safety assessment of a compliant item shall include the procedures, methods, and tools used for assessing the systematic capability along with the justification of its effectiveness to facilitate the re-use of the assessment results in the context of a larger system. {IEC 61508-1, Section 8.2.13.d}

Requirement 5.3.3.11: During the design of safety-related systems, system integration tests shall be planned and documentation of the test planning shall include the test environment, tools, configuration, and programs. {IEC 61508-2, Section 7.4.6.5}

Requirement 5.3.3.12: The safety-related system integration testing shall document the tools and equipment used along with calibration data. {IEC 61508-2, Section 7.5.2.6.d}

Requirement 5.3.3.13: The safety-related system operation and maintenance procedures shall specify the tools necessary for maintenance and revalidation and procedures for maintaining the tools and equipment. {IEC 61508-2, Section 7.6.2.1.g}

Requirement 5.3.3.14: Appropriate documentation of the safety-related system validation testing shall be produced which shall state for each safety function the tools and equipment used, along with calibration data. {IEC 61508-2, Section 7.7.2.4.c}

Requirement 5.3.3.15: The evaluation of the verification results shall contain the configuration of the verification environment and verification tools used, and the calibration data used during the evaluation, if applicable. {ISO 26262-8, Section 9.4.3.2.c}

Requirement 5.3.3.16: The results of tool validation shall be documented. {IEC 61508-3, Section 7.4.4.7; BS EN 50128, Section 6.7.4.5}

Guidance 5.3.3.16.1: The documentation of the results of tool validation should cover the following results: {IEC 61508-3, Section 7.4.4.7; BS EN 50128, Section 6.7.4.5}

- a. a chronological record of the validation activities,
- b. the version of the tool product manual being used,
- c. the tool functions being validated,
- d. tools and equipment used,
- e. the results of the validation activity including a statement that the software has passed the validation or the reason for its failure,
- f. test cases and their results for subsequent analysis, and
- g. discrepancies between expected and actual results.

Requirement 5.3.3.17: The software integration test specification shall state the test environment, tools, configuration, and programs. {IEC 61508-3, Section 7.4.8.2.d; BS EN 50128, Section 7.3.4.37}

Requirement 5.3.3.18: The software/programmable electronics integration test specification (hardware and software) shall state the test environment including tools, support software, and configuration description. {IEC 61508-3, Section 7.5.2.2.d}

Requirement 5.3.3.19: For each safety function, software safety validation shall document the tools and equipment used together with calibration data. {IEC 61508-3, Section 7.7.2.5.d}

Requirement 5.3.3.20: The software architecture specification shall justify that the techniques, measures, and tools chosen form a set which satisfies the software requirements specification at the required software SIL. {BS EN 50128, Section 7.3.4.12}

Requirement 5.3.3.21: The following information shall be available to ensure the proper evaluation or usage of the software tool: {ISO 26262-8, Section 11.4.4.2}

- a. a description of the features, functions, and technical properties of the software tool;

-
- b. the user manual or other usage guides, if applicable;
 - c. a description of the environment required for its operation;
 - d. a description of the expected behavior of the software tool under anomalous operating conditions, if applicable;
 - e. a description of known software tool malfunctions and the appropriate safeguards, avoidance, or work-around measures, if applicable; and
 - f. the measures for the detection of malfunctions and the corresponding erroneous output of the software tool identified during the determination of the required level of confidence for this software tool.

Guidance 5.3.3.21.1: Tool parameters used during the development, verification, or validation of software should be recorded in the development records. {MDEP DICWG-02, Position 8}

Guidance 5.3.3.21.2: If deficiencies exist in the software life cycle data of COTS software tools, the data should be augmented to satisfy requirements. {RTCA DO-178C, Section 2.5.3}

Guidance 5.3.3.21.3: Tool settings used during the development, verification, or validation of baseline equipment, software, or HDL configured devices should be recorded in the development records. This is useful not only for the final software consistency; it also helps in assessing the origin of a fault, which might lie in the source code, in the tool, or in the tool settings. Information about the tool settings used may be critical to assessing the potential for CCFs due to software tools. {IAEA DS-431, Sections 7.163 and 7.164}

Requirement 5.3.3.22: The description of the usage of a software tool shall contain the following information: {ISO 26262-8, Section 11.4.5.1}

- a. the intended purpose;
- b. the inputs and expected outputs; and
- c. the environmental and functional constraints, if applicable.

Requirement 5.3.3.23: The qualification of the software tool shall be documented including the following: {ISO 26262-8, Section 11.4.6.2}

- a. the unique identification and version number of the software tool;
- b. the maximum TCL for which the software tool is classified together with a reference to its evaluation analysis;
- c. the pre-determined maximum ASIL, or specific ASIL, of any safety requirement which might be violated if the software tool is malfunctioning and produces corresponding erroneous output;

- d. the configuration and environment for which the software tool is qualified;
- e. the person or organization who carried out the qualification;
- f. the methods applied for its qualification;
- g. the results of the measures applied to qualify the software tool; and
- h. the usage constraints and malfunctions identified during the qualification, if applicable.

5.3.4 Software Tool Selection and Use

Requirement 5.3.4.1: The software tools that might introduce faults or overlook existing faults in software or in system design shall be selected and used according to documented procedures and rules aiming at reducing this risk. {IEC 62138 Ed. 1.0B, Clause 6.1.4}

Guidance 5.3.4.1.1: Application-oriented development tools that enable plant or system engineers to specify their requirements using graphical techniques and automatically translate the graphical programs into executable application software should be used to reduce the risk of faults when these tools are of adequate quality. {IEC 62138 Ed. 1.0B, Clause 4.1}

Guidance 5.3.4.1.2: The equipment families used for the development of an I&C system should be associated with software tools that can reduce the risk of introducing faults in new application software, in the configuration of pre-developed software, and in the design of the system. {IEC 62138 Ed. 1.0B, Clause 5.1.4}

Guidance 5.3.4.1.3: Methods and tools should be chosen that provide error prevention during the software development process. {RTCA DO-178C, Section 4.2.b.}

Guidance 5.3.4.1.4: The introduction of unnecessary or extraneous requirements by tools that may result in inclusion of unwanted or unnecessary code should be prevented in the requirements phase. {RG 1.152, Section 2.2.2}

Guidance 5.3.4.1.5: Measures should be taken to prevent the introduction of unnecessary design features by tools that may result in unwanted or unnecessary code in the design phase. {RG 1.152, Section 2.3.2}

Requirement 5.3.4.2: The software tool shall be used in a manner such that defects not detected by the software tool will be detected by V&V activities. {IEEE 7-4.3.2-2003, Clause 5.3.2.b; IEEE 7-4.3.2 (draft), Clause 5.3.2.a}

Requirement 5.3.4.3: When a tool or tool version that has the potential to introduce faults in software or in system design is substituted with another, reasonable precautions shall be taken to ensure that this does not have adverse effects on the correctness of the software and the system design. {IEC 62138 Ed. 1.0B, Clause 6.1.4}

Requirement 5.3.4.4: Criteria for the selection and evaluation of tools for the software engineering environment shall be developed and prioritized to allow trade-offs prior to use. {IEC 60880 Ed.2, Clause 14.3.1.2}

Guidance 5.3.4.4.1: A well-known tool with extensive operating experience is preferable to an untried tool with little or no operating experience as long as the well-known tool meets other selection criteria. {IEC 62138 Ed. 1.0B, Clause 5.1.4.1}

Guidance 5.3.4.4.2: The compatibility of a tool with other tools in use should be considered along with the tool quality. {IEC 62138 Ed. 1.0B, Clause 5.1.4.1}

Guidance 5.3.4.4.3: Criteria for the selection and evaluation of tools should be structured by software quality characteristics including functionality, reliability, usability, efficiency, modifiability, portability, licensing effort, resources required to use a tool, the rigor of the quality plan under which the tool was developed, vendor tool history, and alternatives to tool use. {IEC 60880 Ed. 2, Clause 14.3.1.2}

Guidance 5.3.4.4.4: Integrated development environments (IDE), editors, compilers, and linkers should be easy to learn, well integrated or easy to integrate with other tools, default to enforcing standards, and well documented. {NASA-GB-8719.13, Section 11.2}

Guidance 5.3.4.4.5: Activities for the selection of software development environment methods and tools include: {RTCA DO-178C, Clause 4.4.1}

- a. The use of tools or combinations of tools and parts of the software development environment should be chosen to achieve the necessary level of confidence that an error introduced by one part would be detected by another. An acceptable environment is produced when both parts are consistently used together.
- b. If approval is sought for use of the tools in combination, the sequence of the options should be examined and specified in the appropriate plan.
- c. If optional features of software tools are chosen for use in a project, the effects of the options should be examined and specified in the appropriate plan. This is especially important for compilers and autocode generators.
- d. Known tool problems and limitations should be assessed and those issues that can adversely affect software should be addressed.

Guidance 5.3.4.4.6: Methods, techniques, and tools that produce results that cannot be verified to an acceptable degree or that are not compatible with safety requirements should be prohibited, unless analysis shows that the alternative would be less safe. {NUREG-0800 BTP 7-14, Section B.3.1.2.3}

Guidance 5.3.4.4.7: Tools should be used to support all aspects of the I&C life cycle where benefits result through their use and where tools are available. {MDEP DICWG-02, Position 1}

Guidance 5.3.4.4.8: The benefits and risk of using a tool should be balanced against the benefits and risk of not using a tool (i.e., choose tools that limit the opportunity for making errors and introducing faults, but maximize the opportunity for detecting faults). {MDEP DICWG-02, Position 2}

Guidance 5.3.4.4.9: Software tools should be evaluated and selected in accordance with the processes described in IEEE 14102-2010. {IEEE 14102-2010, Clause 10}

Guidance 5.3.4.4.10: Tools should be used to support all aspects of the I&C development life cycle where benefits result through their use and where tools are available. {IAEA DS-431, Section 7.148}

Guidance 5.3.4.4.11: The benefits and risk of using a tool should be balanced against the benefits and risk of not using a tool. The important principle is to choose tools that limit the opportunity for making errors and introducing faults, but maximize the opportunity for avoiding or detecting faults. {IAEA DS-431, Sections 7.152 and 7.153}

Guidance 5.3.4.4.12: Tools should be selected to remain available throughout the system's service life and be compatible with other tools used during system development. {IAEA DS-431, Section 7.154}

Guidance 5.3.4.4.13: The functionality and limits of applicability of all tools should be identified and documented. {IAEA DS-431, Section 7.155}

Guidance 5.3.4.4.14: The tools and their output should not be used outside their declared functionality or limits of application without prior justification. For example, tools cannot replace humans when judgment is involved. In some cases, tool support is more appropriate than complete automation of a process. {IAEA DS-431, Sections 7.156 and 7.157}

Guidance 5.3.4.4.15: Well established methods, languages, and tools for software development should be used. Methods, languages, and tools that are still at the research stage should not be used. {IAEA NS-G-1.1, Section 3.29}

Guidance 5.3.4.4.16: Tools should be selected to facilitate the proper application of the methods, standards, and procedures selected. {IAEA NS-G-1.1, Section 4.6}

Guidance 5.3.4.4.17: A suitable set of tools should be selected for software implementation including translators, code generators, debuggers, linkers, testing, and other verification tools. {IAEA NS-G-1.1, Section 9.11}

Requirement 5.3.4.5: Appropriate tools shall be used to ensure or verify that the right executable code is loaded correctly in the target system. {IEC 60880 Ed.2, Clause 14.3.6.3}

Guidance 5.3.4.5.1: The use of compiler optimization should be avoided. It shall not be used if it produces object code that is excessively difficult to understand, debug, test, and validate. {IEC 60880 Ed. 2, Clause 14.3.4.3}

Guidance 5.3.4.5.2: Appropriate tools should be used to test and/or simulate the behavior of the executable code loaded in the target system. {IEC 60880 Ed. 2, Clause 14.3.6.2}

Guidance 5.3.4.5.3: The use of software tools should be evaluated in the overall context of the quality control and V&V process, and there should be a method of evaluating the output of the tool. {NUREG-0800 Appendix 7.1-D, Section 5.3.2}

Guidance 5.3.4.5.4: Tools and their output should not be used outside their declared functionality or limits of application without prior justification. {MDEP DICWG-02, Position 4}

Guidance 5.3.4.5.5: The selection, qualification, and use of software tools for the development of software for safety systems should follow the guidelines of Section 14 of IEC 60880. {MDEP DICWG-02, Position 9}

Guidance 5.3.4.5.6: Validated tools should be used since they relieve programmers from tasks of the code production process that are prone to manual error, such as programming and verification. {IAEA NS-G-1.1, Section 9.10}

Guidance 5.3.4.5.7: Thoroughly tested translators that are available and maintainable throughout the system lifetime should be used. If no thoroughly tested translator is used, additional analysis and verification, manual or by the use of other tools, should demonstrate that the translation is correct. {IAEA NS-G-1.1, Section 9.12}

Requirement 5.3.4.6: Technical independence requires the independent V&V effort to use its own set of test and analysis tools separate from the developers' tools. {IEEE 1012-2012, Annex C.1}

Guidance 5.3.4.6.1: Different tools should be used on redundant systems to reduce the possibility of common-mode failure. {NUREG/CR-6263, Section 14.2.3}

Guidance 5.3.4.6.2: The teams performing V&V should be independent of the development team using different techniques and tools. {IAEA NS-G-1.1, Section 4.17}

Guidance 5.3.4.6.3: Diversity of methods, languages, tools, and personnel should be taken into consideration to reduce the potential for software common-cause failures. {IAEA NS-G-1.1, Section 3.13}

Requirement 5.3.4.7: Where applicable, automatic testing tools and integrated development tools shall be used to avoid systematic faults. {IEC 61508-2, Section 7.4.6.4}

Guidance 5.3.4.7.1: Special care should be taken when reviewing software tool usage, especially compilers and linkers, to ensure that the single-failure criterion is not violated. For example, diverse systems that have been compiled with a single compiler may subject the code output to a common-mode failure mechanism. {NUREG-0800 Appendix 7.1-D, Section 5.1}

Guidance 5.3.4.7.2: A suitable set of implementation tools should be selected with the aim of minimizing error. {IAEA DS-431, Section 9.58}

Guidance 5.3.4.7.3: Any tools used in the software implementation phase should be compatible with other tools used in other phases of the development. {IAEA NS-G-1.1, Section 9.13}

Requirement 5.3.4.8: Modifications to safety-related systems or software maintenance shall be performed with at least the same level of expertise, automated tools, planning, and management as the initial development of the safety-related system or software. {IEC 61508-2, Section 7.8.2.3; BS EN 50128, Section 9.2.4.17}

Requirement 5.3.4.9: Software off-line support tools shall be selected as a coherent part of the software development activities. {IEC 61508-3, Section 7.4.4.2; BS EN 50128, Section 6.7.4.1}

Guidance 5.3.4.9.1: Software tools should support the development activities that contribute to the correctness of software and system design. {IEC 62138 Ed. 1.0B, Clause 5.1.4}

Guidance 5.3.4.9.2: Success in achieving systematic safety integrity depends on selecting techniques with attention to the consistency and the complementary nature of the chosen methods, languages, and tools for the whole development cycle; whether the developers use methods, languages, and tools they fully understand; and whether the methods, languages, and tools are well-adapted to the specific problems encountered during development. {IEC 61508-3, Section 7.1.2.7}

Guidance 5.3.4.9.3: Appropriate off-line tools to support the development of software should be used in order to increase the integrity of the software by reducing the likelihood of introducing or not detecting faults during the development. Examples of tools relevant to the phases of the software development life cycle include: {IEC 61508-3, Section 7.4.4.2; BS EN 50128, Section 6.7.4.1}

- a. transformation or translation tools that convert a software or design representation (e.g., text or a diagram) from one abstraction level to another level: design refinement tools, compilers, assemblers, linkers, binders, loaders, and code generation tools;
- b. V&V tools such as static code analyzers, test coverage monitors, theorem proving assistants, and simulators;

- c. diagnostic tools used to maintain and monitor the software under operating conditions;
- d. infrastructure tools such as development support systems;
- e. configuration control tools such as version control tools; and
- f. application data tools that produce or maintain data which are required to define parameters and to instantiate system functions including function parameters, instrument ranges, alarm and trip levels, output states to be adopted at failure, and geographical layout.

Guidance 5.3.4.9.4: Off-line support tools should be selected to be integrated so they work co-operatively such that the outputs from one tool have suitable content and format for automatic input to a subsequent tool, thus minimizing the possibility of introducing human error in the reworking of intermediate results. {IEC 61508-3, Section 7.4.4.2}

Guidance 5.3.4.9.5: Off-line support tools should be selected to be compatible with the needs of the application, of the safety-related system, and of the integrated toolset. {IEC 61508-3, Section 7.4.4.2}

Guidance 5.3.4.9.6: The availability of suitable tools to supply the services that are necessary over the whole lifetime of the safety-related system should be considered. {IEC 61508-3, Section 7.4.4.2}

Guidance 5.3.4.9.7: Consideration should be given to the competence of the users of the selected tools. {IEC 61508-3, Section 7.4.4.2}

Guidance 5.3.4.9.8: All tools used should be mutually compatible. {IAEA NS-G-1.1, Section 3.25}

Requirement 5.3.4.10: The selection of the off-line support tools in classes T2 and T3 shall be justified. {IEC 61508-3, Section 7.4.4.4; BS EN 50128, Section 6.7.4.2}

Requirement 5.3.4.11: All off-line support tools in classes T2 and T3 shall have a specification or product documentation that clearly defines the behavior of the tool and any instructions or constraints on its use. {IEC 61508-3, Section 7.4.4.4; BS EN 50128, Section 6.7.4.3}

Requirement 5.3.4.12: The software development process for the software item, including life cycle phases, methods, languages, and tools, shall be consistent across all the sub-phases of the software life cycle and be compatible with the system and hardware development phases, such that the required data can be transformed correctly. {ISO 26262-6, Section 5.4.4}

Guidance 5.3.4.12.1: Corresponding tools should be selected for each sub-phase of software development including guidelines for their application. {ISO 26262-6, Section 5.4.5}

5.3.5 Software Tool Development, Qualification, and Dedication

Requirement 5.3.5.1: Validation that a tool performs its required functions and operates correctly within the software system design shall be accomplished. {FAA-STD-026A, Clause 4.4}

Guidance 5.3.5.1.1: Tools used to analyze the consistency and completeness of the computer system requirements should be validated to a level of confidence proven equivalent to the level required from the design process. {IAEA NS-G-1.1, Section 5.9}

Requirement 5.3.5.2: Qualification of a tool is only required if one of the system or software life cycle processes is eliminated, reduced, or automated and the tool has not been verified. {FAA Order 8110.49 and FAA Order 8110.49 Change 1, Chapter 9-1}

Requirement 5.3.5.3: If a tool eliminates, reduces, or automates software life cycle processes and its output is not verified, the tool shall be qualified. {RTCA DO-178C, Section 12.2.3}

Guidance 5.3.5.3.1: A comprehensive test suite is an effective approach to increase objectivity and uniformity in the application of the qualification criteria to structural coverage analysis tools. {DOT/FAA/AR-06/54, Section 2}

Guidance 5.3.5.3.2: Some compilers have features intended to optimize performance of the object code. If the test cases give coverage consistent with the software level, the correctness of the optimization need not be verified. Otherwise, the impact of these features on structural coverage analysis should be determined. {RTCA DO-178C, Clause 4.4.2}

Guidance 5.3.5.3.3: When user-modifiable software is included in the project, the non-modifiable component should be protected from the modifiable component to prevent interference in the safe operation of the non-modifiable component. This protection can be enforced by hardware, by software, by the tools used to make the change, or by a combination of the three. If the protection is provided by a tool, the tool should be qualified. {RTCA DO-178C, Section 5.2.3}

Guidance 5.3.5.3.4: Changes to an application or development environment should be identified, analyzed, and re-verified. If a new development environment uses software tools, the tool may need to be qualified. {RTCA DO-178C, Section 12.1.3.a}

Guidance 5.3.5.3.5: Using a different autocode generator or a different set of autocode generator options may change the source code or object code generated. Therefore, the impact of these changes should be analyzed. {RTCA DO-178C, Section 12.1.3.c}

Guidance 5.3.5.3.6: If a different compiler or options are used, previous verification may not be valid and should not be used for the new application. {RTCA DO-178C, Section 12.1.3.d}

Guidance 5.3.5.3.7: When automatic code generation (ACG) tools are used in the software development process to eliminate, reduce, or automate life cycle processes, the

following guidelines should be used when applicants are qualifying ACG tools. {Position Paper CAST-13, Section 3}

- a. The software level of the tool needs to be determined. Where there is no justification to reduce the software level of the tool, the tool should be qualified to the same software level and objectives as that of the safety-related plant software.
- b. If the tool is designed and developed in such a way that the ACG tool's architecture and logic can produce unintended, non-deterministic, or erroneous code, then a software level reduction is not possible.
- c. A software level reduction is possible if it can be demonstrated that potential errors in the ACG tool's output can be detected by additional verification on the code produced or if it can be demonstrated that potential errors and failure conditions in the software as a result of the ACG tool's software level reduction or objective's alleviation cannot contribute to the failures of the software.
- d. The applicant can propose a reduction in the ACG software level based on a safety assessment, software architecture, tool partitioning, additional verification, and/or by other means.
- e. If justification of a lower software level exists, the verification objectives and activities for the reduced software level should be satisfied. Justification of a lower software level should address the consequences of a tool's software level reduction on the safety-related plant software.
- f. Some life cycle process objectives can be alleviated or removed if it can be demonstrated that incompatibility problems with the ACG tool are obvious and readily detected and the applicants supply relevant rationale and justification for each objective's alleviation.

Guidance 5.3.5.3.8: Requirements for tools should meet the following characteristics: (RTCA DO-330, Section 5.2.1.2)

- a. should be developed and documented in a manner that provides a determination if the tool satisfies the tool operational requirements;
- b. should include all functional and interface-related requirements that will be implemented in the tool;
- c. should be developed following the processes described in the tool development plan;
- d. should be developed using the tool requirements standards;
- e. should be verifiable and consistent;

- f. should be defined such that abnormal behavior is detected and that invalid output is prevented;
- g. each requirement should trace to one or more tool operational requirements, with the exception of derived tool requirements;
- h. derived tool requirements (i.e., those not traceable to tool operational requirements) should be justified, and they should be evaluated to ensure that they do not negatively impact the expected functionality and outputs defined in the tool operational requirements;
- i. should provide user instructions, list of error messages, and constraints;
- j. should identify requirements related to unused tool functions that do not trace to tool operational requirements; and
- k. should be defined to a level of detail appropriate to ensure proper implementation and to assess correctness of the tool.

Guidance 5.3.5.3.9: The tool design process should define the tool architecture. (RTCA DO-330, Section 5.2.2.2.a)

Guidance 5.3.5.3.10: The tool design should address all tool architecture features, such as protection, if applicable. Protection may be used to isolate parts of the tool or the collection of tools to apply a different approach or different level of qualification. {RTCA DO-330, Section 5.2.2.2.b}

Guidance 5.3.5.3.11: The tool requirements should be refined into low-level tool requirements that are traceable to tool requirements or identified as derived low-level requirements. {RTCA DO-330, Section 5.2.2.2.c}

Guidance 5.3.5.3.12: The tool design description, including tool architecture and low-level tool requirements, should conform to the tool design standards. {RTCA DO-330, Section 5.2.2.2.e}

Guidance 5.3.5.3.13: The tool source code should implement the low-level tool requirements and conform to the tool architecture. {RTCA DO-330, Section 5.2.3.2.a}

Guidance 5.3.5.3.14: The tool source code should conform to the tool code standards. {RTCA DO-330, Section 5.2.3.2.b}

Guidance 5.3.5.3.15: The tool source code should be traceable to the low-level tool requirements. {RTCA DO-330, Section 5.2.3.2.c}

Guidance 5.3.5.3.16: Inadequate or incorrect inputs detected during the tool coding process should be provided to the process(es) that produced the incorrect input(s) for clarification or correction. {RTCA DO-330, Section 5.2.3.2.d}

Guidance 5.3.5.3.17: The tool executable object code should be generated from the tool source code and the compiler and linking instructions. It should be noted that the tool executable object code sometimes includes other files, such as configuration files. {RTCA DO-330, Section 5.2.4.2.a}

Guidance 5.3.5.3.18: Inadequate or incorrect inputs detected during the tool integration process should be provided to the process(es) that produced the inadequate or incorrect input(s). {RTCA DO-330, Section 5.2.4.2.b}

Guidance 5.3.5.3.19: If the tool verification environment is different than the tool development environment, or if there are multiple tool verification environments, then the tool executable object code should be installed in the tool verification environment(s). {RTCA DO-330, Section 5.2.4.2.c}

Guidance 5.3.5.3.20: Verification is the technical assessment of the outputs of both the tool development processes and the tool verification process. The tool verification process is applied as defined by the tool planning process and the tool verification plan. Verification includes reviews, analyses, and tests. {RTCA DO-330, Section 6.1}

Guidance 5.3.5.3.21: In order to determine that the higher level of requirements is satisfied by the lower levels of requirements (or source code) and to identify derived requirements, traceability should exist as follows: {RTCA DO-330, Section 6.1.2.a}

- a. tool requirements should be traceable to the tool operational requirements,
- b. the low-level tool requirements should be traceable to the tool requirements,
and
- c. the tool source code should be traceable to the low-level tool requirements.

Guidance 5.3.5.3.22: In order to identify the requirements not covered by test cases, the test cases should be traceable to the requirements (tool requirements and low-level tool requirements). {RTCA DO-330, Section 6.1.2.b}

Guidance 5.3.5.3.23: When it is not possible to verify specific tool requirements, other means should be provided and their justification for satisfying the tool verification process objectives defined in the tool verification plan and/or tool verification results. {RTCA DO-330, Section 6.1.2.d}

Guidance 5.3.5.3.24: Deficiencies and errors discovered during the tool verification process should be reported to the appropriate tool development processes for clarification and correction. {RTCA DO-330, Section 6.1.2.e}

Guidance 5.3.5.3.25: When a project includes reusable verification or development tools, the following general guidelines apply for reusable tool qualification. {Position Paper CAST-22, Section 3.0}

- a. When reusable software verification and development tools are initially planned, the boundaries and limitations of what is reusable and what will be project specific should be established during the first acceptance of the reusable tool qualification package.
- b. The tools should be packaged and qualified in such a way that some of the tool qualification data may be reused on other projects.
- c. The initial reusable tool qualification must be done in the context of an actual certification project and must have the documented agreement of all the stakeholders.
- d. For each objective that the tool automates, replaces, or supplements, the amount of credit being sought (full, partial, or no credit), assumptions, means of compliance, tool limitations, and remaining activities to be completed by the tool user, system developer, and/or applicant must be described in the reusable tool documents.

Requirement 5.3.5.4: The output of software tools used in the life cycle processes of safety-related software shall be confirmed as suitable for use in safety-related systems by one or both of the following methods. {IEEE 7-4.3.2-2010, Section 5.3.2}

- a. The output shall be subject to the same level of V&V as the safety-related software.
- b. The tool shall be developed using the same or equivalent high-quality life cycle process as required for the software upon which the tool is being used or the tool shall be commercially dedicated.

Guidance 5.3.5.4.1: Prevention or detection of software tool erroneous output can be accomplished through process steps, redundancy in tasks, redundancy in software tools, or by rationality checks within the software tool itself. {ISO 26262-8, Section 11.4.5.2}

Guidance 5.3.5.4.2: A test-tool validation program should be developed to provide confidence that the necessary features of the software tool function as required. {IEEE 7-4.3.2-2003, Clause 5.3.2.a}

Guidance 5.3.5.4.3: COTS and pre-developed tools can be qualified under the commercial dedication processes of EPRI-TR-106439. {DI&C-ISG-06, Section D.10.4.2.3.2}

Guidance 5.3.5.4.4: Tools used for software development may reduce or eliminate the ability of the vendor to evaluate the output of those tools. Therefore, subsequent testing should be performed to show the software will perform as intended. {NUREG 0800, Appendix 7.1-D, Section 5.3.2}

Guidance 5.3.5.4.5: Commercial-grade item dedication often involves engineering judgment. The basis for these judgments should be documented and retained as part of the dedication records. The documentation of the judgment should be sufficient to allow

the dedication process, and the basis for the engineering judgments used in the dedication, to be reviewed. The documentation should also be sufficient to allow a comparably qualified individual to reach the same conclusion. {EPRI TR 106439, Page 4-19}

Guidance 5.3.5.4.6: Reasonable assurance of tool performance must be based on facts, actions, or observations. However, the decision that reasonable assurance has been attained is inherently subjective and the judgment of reasonability may vary between different observers. {EPRI TR 102260, Section 2.1.2.2.3}

Guidance 5.3.5.4.7: The primary difference between a critical characteristic for design and a critical characteristic for acceptance is that the critical characteristics for acceptance need only be based on the complexity, intended safety-related function, and performance of the tool. {EPRI TR 102260, Section 2.3.2.4}

Guidance 5.3.5.4.8: System requirements should be developed using a controlled procedure or methodology. Based on the system requirements, a set of critical requirements can then be generated for the commercial-grade dedication process or evaluation of the software tools used. {EPRI TR 107339, Section 5.1}

Guidance 5.3.5.4.9: V&V activities common to software development in digital systems are critical characteristics that can be verified as being performed correctly following the completion of the software development by conducting certain dedication activities such as audits, examinations, and tests. {EPRI TR 1025243, Page viii}

Guidance 5.3.5.4.10: V&V tasks of witnessing, reviewing, and testing are not required for software tools, provided the software that is produced using these tools is subject to V&V activities that will detect flaws introduced by the tools. {RG 1.168, Section C.2}

Requirement 5.3.5.5: Software tools that impact safety-related software shall be confirmed as suitable by one of the following methods. {IEEE 1012-2004, Clause 4; IEEE 1012-2012, Clause 5; and IEEE 1012-1998, Section 7.4.6}

- a. Tools that insert or translate code (e.g., optimizing compilers, auto-code generators) shall be assigned the same integrity level and verified and validated to the same rigor as the highest software integrity level assigned to the software element that the tool affects.
- b. Tools that do not insert code shall be verified and validated to assure that they meet their operational requirements.
- c. If a tool cannot be verified and validated, then the output of the tool shall be subject to the same level of V&V as the software element.
- d. If tools do not insert code and partitioning of tool functions can be demonstrated, only those functions that are used in the V&V processes shall be verified to demonstrate that they perform correctly for their intended use.

Guidance 5.3.5.5.1: Tools should be verified and assessed consistent with the tool reliability requirements, the type of tool, the potential of the tool to introduce faults or fail to make the user aware of existing faults, and the extent to which the tool may affect redundant elements of a system or diverse systems. Examples of situations that can affect the degree of verification and assessment needed include, for example: {IAEA DS-431, Sections 7.158 and 7.159}

- a. tools that have the ability to introduce faults need to be verified to a greater degree than tools that are demonstrated to not have that capability,
- b. tools that can fail to make the user aware of existing faults need to be verified to a greater degree than tools that do not have that capability,
- c. verification is not necessary for tools when the output of the tool is systematically and independently verified, and
- d. less rigor in tool verification may be accepted if there is mitigation of any potential tool faults (e.g. by process diversity or system design).

Guidance 5.3.5.5.2: When automatically generated code is safety critical or resides within an unprotected partition with safety-critical code, the automatically generated code should be subject to the same inspection, analysis, and testing as hand-generated code. {NASA-GB-8719.13, Section 11.3}

Requirement 5.3.5.6: A tool qualification strategy shall be produced and the tools shall be qualified in accordance with that strategy. The strategy shall consider the reliability requirements of the tool and the type of the tool. The reliability requirements of a tool shall be determined considering: {IEC 60880 Ed.2, Clause 14.3.2.1 and Clause 14.3.2.2}

- a. the consequences of a fault in the tool,
- b. the probability that a tool causes or induces faults in the software implementing the safety function,
- c. what other tools or processes mitigate the consequences of a fault in the tool.

Guidance 5.3.5.6.1: Tools should be verified and assessed consistent with the tool reliability requirements, the type of tool, and the potential of the tool to introduce faults. {MDEP DICWG-02, Position 5}

Guidance 5.3.5.6.2: All tools should be qualified to a level commensurate with their function in the development of the software and in the safety demonstration. The techniques used to gain confidence in the tools should be specified and documented. {IAEA NS-G-1.1, Section 3.25}

Requirement 5.3.5.7: Tools used to provide diversity (i.e., compilers used for the development of multiple version dissimilar software systems) shall be demonstrated to be dissimilar. {IEC 60880 Ed. 2, Clause 14.3.1.5}

Guidance 5.3.5.7.1: Demonstrating that tools used to provide diversity are dissimilar may be achieved by showing that: {IEC 60880 Ed. 2, Clause 14.3.1.5}

- a. each tool was obtained from a different supplier (for example, one tool could be developed and the other tool could be purchased off the shelf),
- b. the tools have different input and/or output languages, or
- c. the tools have dissimilar requirements and design processes.

Requirement 5.3.5.8: Where compiler optimization is used, tests, verification and/or validation shall be performed on the optimized code. {IEC 60880 Ed. 2, Clause 14.3.4.4}

Guidance 5.3.5.8.1: The fitness for purpose of a non-trusted compiler can be justified by subjecting the object code produced by the compiler to a combination of tests, checks, and analyses that are capable of ensuring the correctness of the code to the extent consistent with the target SIL and in particular: {BS EN 50128, Section 6.7.4.6}

- a. testing has sufficiently high coverage of the implemented code and any unreachable code by testing has been shown by checks or analyses that the function concerned is executed correctly;
- b. checks and analyses have been applied to the object code and shown to be capable of detecting the types of errors which might result from a defect in the compiler;
- c. no more translation with the compiler has taken place after testing, checking, and analysis; and
- d. if further compilation or translation is carried out, all tests, checks, and analyses will be repeated.

Requirement 5.3.5.9: Tests, verification and/or validation shall be carried out to ensure that any additional code (assembly instructions) introduced by a translator which is not directly traceable to source line statements (e.g., error checking code, error and exception handling code, initialization code) is correct. {IEC 60880 Ed. 2, Clause 14.3.4.6}

Guidance 5.3.5.9.1: The assessment of a translator may be performed for a specific application project, or for a class of applications. In the latter case all necessary information on the tool (specification or product manual) regarding the intended and appropriate use of the tool should be available to the user of the tool. The assessment of the tool for a specific project may then be reduced to checking general suitability of the tool for the project and compliance with the specification or product manual (i.e., proper use of the tool). Proper use might include additional verification activities within the specific project. {IEC 61508-3, Section 7.4.4.10}

Guidance 5.3.5.9.2: A validation suite (i.e., a set of test programs whose correct translation is known in advance) may be used to evaluate the fitness for purpose of a

translator according to defined criteria, which should include functional and non-functional requirements. For the functional translator requirements, dynamic testing may be a main validation technique. If possible, an automatic testing suite should be used. {IEC 61508-3, Section 7.4.4.10}

Requirement 5.3.5.10: The compatibility of the tools of an integrated toolset shall be verified. {IEC 61508-3, Section 7.4.4.9}

Requirement 5.3.5.11: Each new version of an off-line support tool shall be qualified. This qualification may rely on evidence provided for an earlier version if sufficient evidence is provided that: {IEC 61508-3, Section 7.4.4.18; BS EN 50128, Section 6.7.4.11}

- a. the functional differences (if any) will not affect tool compatibility with the rest of the toolset; and
- b. the new version is unlikely to contain significant new, unknown faults.

Guidance 5.3.5.11.1: Evidence that a new version of an off-line support tool is unlikely to contain significant new, unknown faults may be based on: {IEC 61508-3, Section 7.4.4.18; BS EN 50128, Section 6.7.4.11}

- a. a clear identification of the changes made,
- b. an analysis of the V&V actions performed on the new version, and
- c. any existing operational experience from other users that is relevant to the new version.

Requirement 5.3.5.12: The methods listed in Table 5.6 shall be applied for the qualification of software tools classified at TCL3. The methods listed in Table 5.7 shall be applied for the qualification of software tools classified at TCL2. A software tool classified at TCL1 needs no qualification methods. A table entry of (++) means the method is highly recommended while an entry of (+) means the method is recommended. The methods are either recommended or highly recommended depending on the ASIL of the software being developed. {ISO 26262-8, Section 11.4.6.1}

Table 5.6 – Qualification of Software Tools Classified TCL3

Methods		ASIL			
		A	B	C	D
1a	Increased confidence from use	++	++	+	+
1b	Evaluation of the tool development process	++	++	+	+
1c	Validation of the software tool	+	+	++	++
1d	Development in accordance with a safety standard	+	+	++	++

Table 5.7 – Qualification of Software Tools Classified TCL2

Methods		ASIL			
		A	B	C	D
1a	Increased confidence from use	++	++	++	+
1b	Evaluation of the tool development process	++	++	++	+
1c	Validation of the software tool	+	+	+	++
1d	Development in accordance with a safety standard	+	+	+	++

- a. If a software tool is to be qualified using the method of increased confidence from use, then the following requirements shall be met. {ISO 26262-8, Section 11.4.7}
- i. Evidence is provided for the following:
- (a) the software tool has been used previously for the same purpose with comparable use cases and with a comparable determined operating environment and with similar functional constraints,
 - (b) the justification for increased confidence from use is based on sufficient and adequate data which can be obtained through accumulated amount of usage (e.g., duration or frequency),
 - (c) the specification of the software tool is unchanged, and
 - (d) the occurrence of malfunctions and corresponding erroneous outputs of the software tool acquired during previous developments are accumulated in a systematic way.
- ii. The experience from the previous usage of the software tool during given development activities shall be analyzed and evaluated by considering the following information:
- (a) the unique identification and version number of the software tool;
 - (b) the configuration of the software tool;
 - (c) the details of the period of use and relevant data on its use;
 - (d) the documentation of malfunctions and corresponding erroneous outputs of the software tool with details of the conditions leading to them;
 - (e) the list of previous versions monitored, listing the malfunctions fixed in each relevant version; and

-
- (f) the safeguards, avoidance measures, or work-arounds for the known malfunctions, or detection measures for a corresponding erroneous output, if applicable.
 - iii. The increased confidence from use argument shall only be valid for the considered version of the software tool.
 - b. If a software tool is to be qualified using the method of evaluation of the tool development process, then the following requirements shall be met. {ISO 26262-8, Section 11.4.8}
 - i. The development process applied for the development of the software tool shall comply with an appropriate standard.
 - ii. The evaluation of the development process applied for the development of the software tool shall be provided by an assessment of the development of an adequate and relevant subset of the features of the software tool based on an appropriate national or international standard and the proper application of the assessed development process shall be demonstrated.
 - c. If a software tool is to be qualified using the method of validation of the software tool, then the following requirements shall be met. {ISO 26262-8, Section 11.4.9}
 - i. The validation of the software tool shall meet the following criteria:
 - (a) the validation measures shall demonstrate that the software tool complies with its specified requirements,
 - (b) the malfunctions and their corresponding erroneous outputs of the software tool occurring during validation shall be analyzed together with information on their possible consequences and with measures to avoid or detect them, and
 - (c) the reaction of the software tool to anomalous operating conditions shall be examined (e.g., foreseeable misuse, incomplete input data, incomplete update of the software tool, use of prohibited combinations of configuration settings).

Guidance 5.3.5.12.1: The qualification process of a tool should take into account experience from prior use. {MDEP DICWG-02, Position 6}

Guidance 5.3.5.12.2: The verification and assessment of software tools should take into account experience from prior use, including experience of the developers and experience gained from the processes in which the tools are used. {IAEA DS-431, Section 7.160}

5.3.6 Software Tool Quality Assurance

Requirement 5.3.6.1: The QA program shall provide control over activities affecting the quality of the identified structures, systems, and components (SSCs), to an extent consistent with their importance to safety. {10 CFR Part 50, Appendix B, Criterion II}

Guidance 5.3.6.1.1: The choice, verification, and assessment of tools should be justified and documented. {IAEA DS-431, Section 7.161}

Guidance 5.3.6.1.2: The software implementation should be established using a predetermined combination of techniques commensurate with the system's importance to safety, covering languages, tools, coding practices, analysis, review, and testing. {IAEA DS-431, Section 9.43.b}

Requirement 5.3.6.2: The QA program shall take into account the need for special controls, processes, test equipment, tools, and skills to attain the required quality. {10 CFR Part 50, Appendix B, Criterion II}

Requirement 5.3.6.3: In those cases where the QA requirements of an original tool cannot be determined, an engineering evaluation shall be conducted by qualified individuals to establish the requirements and controls. This evaluation shall assure that interfaces, interchangeability, safety, fit, and function are not adversely affected or contrary to applicable regulatory or code requirements. {EPRI TR 102260, Section 1.3.3.2}

Guidance 5.3.6.3.1: COTS systems and tools are likely to be proprietary and generally unavailable for review. It is likely that there is no reliable method to determine security vulnerabilities for operating systems and tools. In such cases, unless such systems or tools are modified by the application developer, the security effort should be limited to ensuring that the features within the operating system or tool do not compromise the secure operational environment that would degrade the reliability of the safety system. {RG 1.152, Section C.2.4.2}

Guidance 5.3.6.3.2: When dealing with COTS software tools, operating experience is often the deciding factor in determining a tool's quality. Additional evidence may be based on tool qualification, validation of the suppliers, and tests. {IEC 62138 Ed. 1.0B, Clause 5.1.4}

Guidance 5.3.6.3.3: Evidence should be provided regarding the quality of the software tools that might introduce faults in software or in system design, and regarding their ability to produce correct results. {IEC 62138 Ed. 1.0B, Clause 5.1.4}

Requirement 5.3.6.4: Where the results of tool validation are unavailable, there shall be effective measures to control failures of the executable safety-related system that result from faults that are attributable to the tool (e.g., generation of diverse redundant code which allows the detection and control of failures of the executable safety-related system as a result of faults that have been introduced into the executable safety-related system by a translator). {IEC 61508-3, Section 7.4.4.8; BS EN 50128, Section 6.7.4.6}

Requirement 5.3.6.5: Translators/compilers shall not remove without warning defensive programming or error-checking features introduced by the programmer. {IEC 60880 Ed. 2, Clause 14.3.4.2}

Requirement 5.3.6.6: Measurement equipment used for testing shall be calibrated appropriately and any tools used for testing shall be shown to be suitable for the purpose. {BS EN 50128, Section 6.1.4.1}

Requirement 5.3.6.7: An assessment shall be carried out for offline tools in classes T2 and T3 to determine the level of reliance placed on the tools, and the potential failure mechanisms of the tools that may affect the executable software. Where such failure mechanisms are identified, appropriate mitigation measures shall be taken (e.g., avoid known bugs, restrict use of the tool functionality, check the tool output, use diverse tools for the same purpose). (IEC 61508-3, Section 7.4.4.5; BS EN 50128, Section 6.7.4.2)

Requirement 5.3.6.8: For each tool in class T3, evidence shall be available that the output of the tool conforms to its specification, or documentation of the output or failures in the output are detected. {IEC 61508-3, Section 7.4.4.6; BS EN 50128, Section 6.7.4.4}

Guidance 5.3.6.8.1: For each tool in class T3, evidence that the output of the tool conforms to its specification or that documentation of the output or failures in the output are detected may be based on: {IEC 61508-3, Section 7.4.4.6; BS EN 50128, Section 6.7.4.4}

- a. a suitable combination of history of successful use in similar environments and for similar applications (within the organization or other organizations),
- b. tool validation,
- c. diverse redundant code which allows the detection and control of failures resulting in faults introduced by a tool, and
- d. other appropriate methods for avoiding or handling failures introduced by tools such as version history or a record of the errors and ambiguities associated with tool use in the environment.

Requirement 5.3.6.9: To the extent required by the SIL, the software or design representation selected shall have a translator which has been assessed for purpose including, where appropriate, assessment against the international or national standards. {IEC 61508-3, Section 7.4.4.10; BS EN 50128, Section 6.7.4.7}

Requirement 5.3.6.10: Where automatic code generation or similar automatic translation takes place, the suitability of the automatic translator for safety-related system development shall be assessed at the point in the development life cycle where development support tools are selected. {IEC 61508-3, Section 7.4.4.14; BS EN 50128, Section 6.7.4.9}

Requirement 5.3.6.11: Safety-related software jointly developed with a supplier requires a development interface agreement that specifies the supporting processes and tools, including interfaces, to assure a development environment compatible with the supplier. {ISO 26262-8, Section 5.4.3.1.g}

Requirement 5.3.6.12: Software tool usage, environmental and functional constraints, and general operating conditions shall comply with the tool's evaluation criteria and qualification. {ISO 26262-8, Section 11.4.3.1}

Guidance 5.3.6.12.1: Software tool usage, environmental and functional constraints, and general operating conditions can be shown to comply with the tool's evaluation criteria or its qualification through the use of identical version and configuration settings for the same use cases together with the same implemented measures for the prevention or detection of malfunctions and their corresponding erroneous output, as documented in the qualification report for the software tool. {ISO 26262-8, Section 11.4.3.1}

Requirement 5.3.6.13: If the tool confidence level evaluation or qualification of a software tool is performed independently from the development of a particular safety-related item or element, the validity of this predetermined tool confidence level or qualification shall be confirmed prior to the software tool being used for the development of a particular safety-related item or element. {ISO 26262-8, Section 11.4.2.1}

Requirement 5.3.6.14: When special purpose software is used for environmental qualification testing, the manufacturer must verify, validate, and control the configuration of the special purpose test software. The test software should be included as part of the test setup conformity conducted before the qualification testing. {FAA Order 8110.49, Section 4-3.a.1}

5.3.7 Software Tool Training

Requirement 5.3.7.1: Procedures should be developed to ensure that all persons involved in any overall system or software life cycle activity, including activities for verification, management of functional safety, and functional safety assessment shall have the appropriate competence including training, technical knowledge, experience, and qualifications relevant to the specific duties that they have to perform and such procedures shall include requirements for the refreshing, updating, and continued assessment of competence. {IEC 61508-1, Section 6.2.13}

Guidance 5.3.7.1.1: Software assurance personnel should be trained in relevant software engineering tools so they stay current with the engineering environment and products they must assure. {NASA-STD-8739.8, Section 5.3.1.3}

Requirement 5.3.7.2: The QA program shall provide for indoctrination and training of personnel performing activities affecting quality as necessary to assure that suitable proficiency is achieved and maintained. {10 CFR Part 50, Appendix B, Criterion II}

5.3.8 Software Tool Configuration Management

Requirement 5.3.8.1: SCM shall maintain accurately and with unique identification all configuration items which are necessary to meet the safety integrity requirements of the safety-

related system including all tools and development environments which are used to create or test, or carry out any action on, the software of the safety-related system. {IEC 61508-3, Section 6.2.3.c}

Guidance 5.3.8.1.1: Contractually developed or qualified commercial software products that are safety system software should be taken under control by a SCM program. {RG 1.169, Section C.4}

Guidance 5.3.8.1.2: Configuration items or controlled documents should include support software used in development, and test code used in testing. {RG 1.169, Section C.6}

Guidance 5.3.8.1.3: A SCM program should take control of tools (i.e., tools should be treated as configuration items). SCM plans should identify that software tools used in the production of safety system software be considered as a configuration item. {RG 1.169, Section C.8}

Guidance 5.3.8.1.4: All items of software development, such as compilers, development tools, configuration files and operating systems, should be under CM control. {IAEA NS-G-1.1, Section 4.20}

Guidance 5.3.8.1.5: All tools should be under appropriate CM. {IAEA DS-431, Section 7.162}

Guidance 5.3.8.1.6: Where automatic or semi-automatic tools are used for the production of documentation, specific procedures may be necessary to ensure effective measures are in place for the management of versions or other control aspects of the documents. {IEC 61508-1, Section 5.2.11}

Guidance 5.3.8.1.7: A version history may provide assurance of maturity of the tool, and a record of the errors and ambiguities that should be taken into account when the tool is used in a new development environment. {IEC 61508-3, Section 7.4.4.6}

Requirement 5.3.8.2: The CM system shall cover software development and the software development environment necessary for the reproducibility of the development and for the maintenance activities including all the tools, translators, data and test file, parameterization files, and supporting hardware platforms. {BS EN 50128, Section 6.5.4.12}

Requirement 5.3.8.3: Where off-line support tools of classes T2 and T3 generate items in the configuration baseline, CM shall ensure that information on the tools is recorded in the configuration baseline. This includes the identification of the tool and its version; the identification of the configuration baseline items for which the tool version has been used; and the way the tool was used (including the tool parameters, options, and scripts selected) for each configuration baseline item so the baseline can be reconstructed. {IEC 61508-3, Section 7.4.4.15}

Requirement 5.3.8.4: Configuration management shall ensure that for tools in classes T2 and T3, only qualified versions are used. {IEC 61508-3, Section 7.4.4.16; BS EN 50128, Section 6.7.4.10}

Requirement 5.3.8.5: Configuration management shall ensure that only tools compatible with each other and with the safety-related system are used recognizing that the safety-related system hardware may also impose compatibility constraints on software tools (e.g., a processor emulator needs to be an accurate model of the real processor electronics). {IEC 61508-3, Section 7.4.4.17}

Guidance 5.3.8.5.1: Software tools are defined by the software planning process and identified in the software life cycle environment configuration index. The following guidance should be used to assign a control category to each tool. Based on the control category, the SCM process objectives specified in Table 5.8 should be completed for that tool. {RTCA DO-178C, Sections 7.3 and 7.5}

- a. Configuration identification should be established for the executable object code (or equivalent) of the tools used to develop, control, build, verify, and load the software.
- b. The SCM process for controlling qualified tools should comply with the objectives associated with control category 1 (CC1) for software development tools and control category 2 (CC2) for software verification tools.
- c. Unless paragraph (b) above applies, the SCM process for controlling the executable object code (or equivalent) of tools used to build and load the software (compilers, assemblers, linkage editors) should comply with the SCM process objectives associated with CC2, as a minimum.

Table 5.8 – SCM Process Objective Control Categories

SCM Process Objective	CC1	CC2
Configuration Identification	•	•
Baselines	•	
Traceability	•	•
Problem Reporting	•	
Change Control – integrity and identification	•	•
Change Control – tracking	•	
Change Review	•	
Configuration Status Accounting	•	
Retrieval	•	•
Protection against Unauthorized Changes	•	•
Media Selection, Refreshing, Duplication	•	
Release	•	
Data Retention	•	•

Guidance 5.3.8.5.2: Tool CM process objectives are: {RTCA DO-330, Section 7.1}

- a. each configuration item and its successive versions are labeled unambiguously so that a basis is established for the control and reference of configuration items;
- b. baselines are defined for further tool life cycle process activity and allow reference to, control of, and traceability between configuration items;
- c. the problem reporting process records non-compliance with tool plans and standards, records deficiencies of outputs of tool life cycle processes, records anomalous behavior of tool products, and ensures resolution of these problems;
- d. change control provides for recording, evaluation, resolution, and approval of changes throughout the life cycle;
- e. change review ensures problems and changes are assessed, problems and changes are approved or disapproved, approved changes are implemented, and feedback is provided to affected processes through problem reporting and change control methods defined during the tool planning process;
- f. status accounting provides data for the CM of tool life cycle processes with respect to configuration identification, baselines, problem reports, and change control;
- g. archival and retrieval ensures that the tool life cycle data associated with the tool product can be retrieved in case of a need to duplicate, regenerate, retest, or modify the tool product; and
- h. tool life cycle environment control ensures that other tools used to produce the tool itself are identified, controlled, and retrievable.

5.3.9 Software Tool Review and Approval

Requirement 5.3.9.1: The tool requirements and the critical characteristics and sub-characteristics of the tool shall be reviewed against the requirements for the safety-related software being approved. {IEEE 14102-2010, Clause 10}

Guidance 5.3.9.1.1: Software tool review criteria should conform to the characteristics described in IEEE 14102-2010. {IEEE 14102-2010, Clause 10}

Guidance 5.3.9.1.2: The documentation of the software tool evaluation should be used by the approving authority as evidence of the tool's acceptability. Particular attention should be given to the weighting factors assigned to the various characteristics and sub-characteristics. {IEEE 14102-2010, Clause 10}

Requirement 5.3.9.2: Applicants and tool developers must discuss the details of each reusable tool qualification project with personnel involved in the approval process. {FAA AC 20-148, Section 2.b}

Requirement 5.3.9.3: Review and approval of software tools is dependent on following all the objectives, executing all activities that satisfy each objective, and developing all the associated life cycle data required by RTCA DO-178C, RTCA DO-330, and all its supplements. {FAA AC 20-115C, Section 6.a and 6.b}

Guidance 5.3.9.3.1: Reviewers should thoroughly evaluate tool usage. Tools used for software development may reduce or eliminate the ability for the vendor to evaluate the output of those tools, and therefore, rely on the tool, or on subsequent testing to show the software will perform as intended. Testing alone can only show that those items tested will operate as intended, and cannot be relied upon to show that no unintended functions exist, or that the software will function in conditions other than those specifically tested. The use of software tools should be evaluated in the overall context of the quality control and V&V process, and there should be a method of evaluating the output of the tool. {NUREG-0800 Appendix 7.1-D, Section 5.3.2}

Guidance 5.3.9.3.2: A liaison process should be used as part of the software development life cycle process to establish communication and understanding between the applicant and the approval authority. The approval authority may review the software life cycle processes and data to assess compliance to applicable requirements. {FAA Order 8110.49, Section 2-1.a}

Guidance 5.3.9.3.3: The amount of approval authority involvement in a software project should be determined as HIGH, MEDIUM, or LOW based on the software level of the project, the amount and quality of review support, the experience and history of the applicant and software developer, service difficulty history, and several other factors. {FAA Order 8110.49, Section 3-2.a and 3-2.b}

Guidance 5.3.9.3.4: The use of unusual tools will increase the level of approval authority involvement. {FAA Order 8110.49, Section 3-2.c}

- a. For software level A or B projects, the use of unusual tools requires at least a HIGH level of approval authority involvement.
- b. For software level C or D projects, the use of unusual tools requires at least a MEDIUM level of approval authority involvement.

Guidance 5.3.9.3.5: Tools are allowed to make changes to the modifiable component of user modifiable software provided the following information is provided for approval: {FAA Order 8110.49 and FAA Order 8110.49 Change 1, Chapter 7-6.a}

- a. plans for controlling tool version,
- b. plans for controlling tool usage,

- c. plans for qualifying or verifying the tool, and
- d. procedures for modifying the tool.

Guidance 5.3.9.3.6: Planning documents should describe how the applicant will have visibility into their suppliers' and sub-tier suppliers' activities including COTS software tool suppliers and vendors. These plans should be submitted for review and approval. {FAA Order 8110.49 Change 1, Section 13-3.b.}

Intentionally Blank

6 Summary and Conclusions

ISL analyzed and compared numerous documents from the aerospace industry, civil aviation industry, commercial nuclear power industry, automotive industry, and railway industry. ISL also analyzed and compared numerous documents from several organizations. All industries require that software tool use be well planned and documented. Some industries require that planning and documentation be part of the overall quality assurance (QA) process. Other industries require software tool planning and documentation as part of the software development life cycle processes, activities, and tasks. An ISL observation, based on the analysis of industry practices, is that software tool planning and documentation should occur during the software development life cycle processes and be verified through audits and reviews as part of the QA process.

The coverage of software tool requirements and guidance throughout the various industries varies considerably. Only a few industries impose requirements and provide guidance for software tool use in all phases of the software development process. Most industries do not impose requirements or provide guidance for all types of tools. Most industries focus their efforts on the coding and testing processes and modeling, construction, and implementation tools which can have a direct impact on the development of the safety-related software through the introduction of defects into the executable or failure to detect defects through verification testing. Only a few industries specify software tool requirements and provide guidance for other software development processes like the requirements specification phase, the conceptual design phase, and the detailed design phase. Few industries specify requirements or provide guidance for management support tools, maintenance tools, documentation tools, configuration management tools, and quality assurance tools other than requiring that such tools be identified and their use documented. An ISL observation, based on the analysis of industry practice, is that all industries need to expand coverage of software tool requirements to cover software tool use in all phases of software development and all types of software tools to minimize the potential hazards of using tools.

The overwhelming consensus of industry practice with respect to software tool verification, validation, and qualification is that such activities are only required if the software tool output is not systematically verified (i.e., the output is verified every time the tool is used), if the software tool automates, eliminates, or reduces activities and tasks required by software development life cycle processes, and if the software tool is being used to develop safety-related software. Most industries utilize a graded software classification scheme to determine the rigor applied to the software development process. Some of those industries use the software classification scheme along with the type of software tool to determine the rigor necessary to verify, validate, or qualify software tools used in the software development processes. Only the civil aviation industry has published a document (i.e., RTCA DO-330) that specifically covers software tool development and qualification including qualification of commercial off-the-shelf (COTS) software tools. The automotive industry (i.e., ISO 26262) and the commercial nuclear power industry (i.e., IEC 60880) recommend methods of software tool qualification and provides guidance for those methods but lacks specific procedures for implementing the methods. An ISL observation, based on an analysis of tool qualification practices, is that tool qualification

would not fit well into the NRC regulatory framework. If the software commercial-grade dedication process of EPRI TR-1025243 is expanded to cover all software tools, it would only be a licensee or applicant acceptance process of the software tool through special tests and inspections, a survey of the commercial-grade supplier, source verification, or supplier and item performance history. Tool qualification could be made a critical characteristic of a software tool and then tool qualification subjected to the commercial-grade dedication process. In this case, regulatory guidance would need to adopt an industry consensus practice for software tool qualification to support the critical characteristic designation.

The following sections summarize the analysis of each industry and organization and presents ISL conclusions of each industry's practice based on the analysis.

6.1 Aerospace Industry

National Aeronautics and Space Administration (NASA) documents identify and recommend tools for many of the software engineering processes and define QA requirements for software tools. NASA uses a software classification system that is unique to the industry in that embedded system software, support software, software tools, and routine office software are all covered by the single classification system. NASA requires that an independent QA organization classify software tools to verify proper classification.

Similar to other industries, NASA documents state that software tools used in developing software used in safety-critical systems should be identified and the level of rigor associated with the verification, validation, and accreditation of software tools should be determined by the tool functions and the safety classification of the systems in which the software will be used. However, NASA does not provide detailed procedures for tailoring the QA requirements, developing software tools, or verifying, validating, accrediting, and qualifying software tools based on tool function and system characteristics.

NASA-STD-7009 provides a comprehensive process for developing, verifying, and validating models, simulations, and analysis tools. However, NASA does not provide specific verification, validation, or qualification requirements for most other commercial off-the-shelf (COTS), government off-the-shelf (GOTS), or modified off-the-shelf (MOTS) software tools except that automatically generated code must be treated to the same level as hand-generated code.

NASA implements a detailed compliance checklist for supporting software review and approval; however, the checklist does not apply to software tools in its current form. Although other industries contain more comprehensive processes for software tool development, verification, validation, qualification, review, and approval, the adoption of the key elements of NASA-STD-7009 and use of a requirements compliance matrix would form a good technical basis and should be considered in any new regulatory guidance for the commercial nuclear power industry.

6.2 Civil Aviation Industry

Federal Aviation Administration (FAA) and Radio Technical Commission for Aeronautics (RTCA) documents provide comprehensive, well-organized software tool requirements,

guidance, and review and approval practices that would form a solid technical basis for developing new regulatory guidance for the commercial nuclear power industry.

Consistent with other industries, the FAA requires that all software tools be identified, described, and validated, but does not supply any guidance or define any processes for the validation of tools. The identification and description of software tools is addressed through specific documentation requirements that define the format and content of documents that discuss data generated during the software development process and detailed information on software tools used during the software development process.

RTCA DO-178C defines when software tools require qualification and the level of rigor necessary for the qualification based on a two-dimensional, deterministic process of assigning a software level based on a safety assessment of the system containing the software and performing a tool impact assessment to determine the type of tool and the tool's impact on the software development life cycle processes. Risk and probability are not used in the FAA tool qualification process. However, consistent with NRC practices of risk-informed regulation, NRC regulatory guidance on software tools could address the use of risk insights, in terms of the values of risk importance measures that have been determined for various plant structures, systems, and components (SSCs), some of which involve digital instrumentation and control (DI&C) systems, equipment, and software, in conjunction with deterministic methods (e.g., failure modes and effects analyses (FMEAs)) to help determine the importance to safety of the potentially affected SSCs and the safety significance of the credible modes of failure of the potentially affected SSCs. Such insights might be useful as an input to the process of determining the rigor needed for qualification of various types of software tools.

The primary method of tool qualification in accordance with RTCA DO-330 is development, verification, and validation in accordance with a high-quality and well-organized software tool development life cycle process. Alternative tool qualification methods include using service history, exhaustive input testing, formal methods, and use of dissimilar tools. Unlike most other industries that superficially address COTS software tools, RTCA DO-330 provides a comprehensive qualification process for COTS tools that divides qualification responsibilities between the COTS developer and tool user. RTCA DO-330 contains a tool qualification liaison process that contains similar activities and tasks as Interim Staff Guidance (ISG) 06 to facilitate the review and approval of software tools. FAA Order 8110.49 describes the types of reviews performed by the certification authority, the tool qualification plans and tool qualification data that needs to be provided for these reviews, and defines the level of FAA involvement in software projects. Neither FAA Order 8110.49 nor RTCA DO-330 contain a requirements compliance checklist to facilitate the review and certification process comparable to the aerospace industry.

Although the civil aviation industry would form a sound technical basis for developing commercial nuclear power regulatory guidance, changes to the guidance would be beneficial to provide a more comprehensive and practical process for software tool qualification, review, and approval for the commercial nuclear power industry. The changes would involve enhancing the verification tool qualification requirements associated with the most safety-critical software and

modifying COTS software tool qualification to follow a more practical approach of either determining the adequacy of the vendor's proprietary process or use an alternate method of qualification such as exhaustive input checking or the use of dissimilar tools.

One possible approach for the NRC might be that FAA AC 20-115C could be viewed as being analogous to an NRC regulatory guidance document similar to BTP 7-14, ISG-06, or a regulatory guide, probably developed in that order if developed, in which the NRC could adopt the industry practices discussed in RTCA DO-330 and its supplements, RTCA DO-331 through 333, as well as others found to be useful through ISL's analyses as discussed in this report. In Task 3, ISL will be developing the technical and regulatory basis for adapting and adopting the most appropriate methods in use for licensee or applicant software tool acceptance, which we expect will most likely be done by means of commercial-grade dedication.

6.3 Nuclear Regulatory Commission

NRC regulatory guides (RGs), standard review plan (SRP) branch technical positions (BTPs) and appendices, ISG, and contractor reports provide high-level software tool guidance primarily through endorsement or adoption of Institute of Electrical and Electronic Engineers (IEEE) standards. Regulatory guidance in the areas of software tool commercial-grade dedication, qualification, development, review, and approval could be improved.

The NRC uses a two-level safety classification system of safety-related and nonsafety-related. However, unlike most industries that use multiple safety integrity levels (SILs) to determine the minimum software verification, validation, or QA activities for safety-related software, the NRC does not endorse that type of scheme for safety-related software and instead requires that safety-related software used in a commercial nuclear power plant be assigned the highest SIL. Nonsafety-related software executed on the same computer as safety-related software also must be treated as safety-related and assigned the highest SIL.

RG 1.152 discusses several documents (i.e., IEEE 7-4.3.2-2003, 10 CFR 50.55a(h), IEEE 279-1971, and IEEE 603-1991) that address barriers between safety and nonsafety functions in digital computers used in nuclear power plants. IEEE 7-4.3.2-2003 does not require nonsafety software to meet the requirements of IEEE 7-4.3.2-2003 if barriers between safety and nonsafety software are designed in accordance with the requirements of IEEE 7-4.3.2-2003 and do not interfere with the performance of the safety functions of the safety software or firmware. In its qualified endorsement of IEEE 7-4.3.2-2003, RG 1.152 takes exception to the IEEE 7-4.3.2-2003 position on nonsafety software and clarifies that any software providing nonsafety functions that resides on a computer providing a safety function must be classified as a part of the safety system. An ISL observation, based on an analysis of RG 1.152 and the barrier and separation requirements of IEEE 7-4.3.2-2003, is that the barrier and separation requirements of IEEE 7-4.3.2-2003 have no relevance to software tools and are only applicable to system hardware and system software. An additional ISL observation is that the treatment of nonsafety-related software is inconsistent with the classification and commercial-grade dedication of safety-related and nonsafety-related software based on its importance to safety as described in 10 CFR Part 50, Appendix B, and 10 CFR Part 21.

IEEE 603-2009 states that equipment performing nonsafety functions that is connected to safety-related equipment should either be classified as an associated circuit or shall be electrically isolated from the safety system, have digital communication independence, and be classified as non-Class 1E. The RG 1.152 exception to the IEEE 7-4.3.2-2003 position on nonsafety software is based on IEEE 603-1991 requirements for barrier and separation of safety and non-safety software which are not consistent with the IEEE 603-2009 requirements for nonsafety software that is connected to safety software. The RG 1.152 position also negates the benefits of software partitioning which can isolate the nonsafety functions from the safety functions to minimize the cost and manpower required to qualify the nonsafety portions of the system because nonsafety systems do not require commercial-grade dedication for use in commercial nuclear power plants. An ISL observation, based on our analysis, is that barrier and separation requirements discussed in RG 1.152 should be updated to address the IEEE 603-2009 changes to the requirements for nonsafety-related systems, and should address an approach to software classification and dedication based on the importance of the software to the safety of the system.

The main focus of the regulatory positions in RG 1.152 is to provide specific guidance for establishing and maintaining a secure development and operational environment for the protection of digital safety systems throughout the design and development life cycle phases of the safety system. RG 1.152 promulgates regulatory positions for five phases of system development including the concept phase, requirements phase, design phase, implementation phase, and testing phase. An ISL observation, based on an analysis of RG 1.152 guidance, is that the need for establishing and maintaining a secure operating and development environment for system software is not relevant to software tools because neither the operational environment nor development environment of the software tool needs to be secure even if the software tool is used to develop safety-related software because tools are not likely to be hacked while being developed and a would-be attacker of the application software would not know which tool(s) was used or would be used in its development.

RGs 1.168 through 1.173 contain some of the common industry requirements for software tool identification, configuration control, documentation, and verification and validation (V&V) through endorsement of various IEEE standards. RGs 1.168 through 1.173 lack guidance for software tool review and approval similar to the certification liaison process in RTCA DO-330. The regulatory guides are primarily written to address safety-related software development and do not address software tools beyond the simple high-level requirements.

RG 1.168 takes exception to the graduated SIL classifications of IEEE 1012-2004 and states that software used in nuclear power plant safety systems should be assigned a SIL 4 classification. An ISL observation, based of this regulatory position, is that classifying all safety-related software as SIL 4 does not allow the V&V activities for safety-related software development to be tailored based on the importance to safety of the various elements of the safety-related system as specified in 10 CFR Part 50, Appendix B, Criterion II, "Quality Assurance Program."

RG 1.173 endorses IEEE 1074-2006 as a standard for developing software life cycle processes, and each organization that uses this standard must map the activities specified in the standard into its own software life cycle. However, IEEE 1074-2006 does not address system-level issues. Therefore, the overall system-level development processes must be developed in accordance with other regulatory guidance or industry standards. ISL recommends that the NRC consider revising RG 1.173 to endorse IEEE 12207-2008 and IEEE 15288-2008, which contain software and system development processes, respectively, and have IEEE, ISO, and IEC endorsement. Life cycle processes are relevant to software tools because the requirements and guidance for the selection, use, verification, validation, qualification, review, and approval of tools vary within each of the different life cycle processes. A thorough discussion of IEEE 12207-2008 and IEEE 15288-2008 is contained in Section 2 of ISL-ESD-TR-13-05 which is the Task 1 report from this research project.

ISG-06 provides guidance for two methods of approving a component for use in safety-related applications. The first method is to commercially dedicate a commercial-grade item (CGI) and the second method is to design and manufacture the component under a 10 CFR Part 50, Appendix B-compliant QA program. Software tools are not mentioned in ISG-06 with respect to the approval processes; however, the two approval methods could be applied to software tools.

BTP 7-14 provides the acceptance criteria for NRC review of planning documentation associated with the development of safety-related software. BTP 7-14 acceptance criteria for planning documents includes many high-level requirements for software tools. Planning documents for most phases of the software development life cycle are required to describe tools used in that life cycle phase, define the process by which tools are selected, define the tool operational environment and any special controls necessary to execute the tools. In some cases, the planning document is required to contain a tool qualification plan. However, BTP 7-14 does not provide or reference any specific guidance for qualifying the tool or developing the tool similar to RTCA DO-178C and RTCA DO-330, International Electrotechnical Commission (IEC) Standards 61508 and 60880, International Organization for Standardization (ISO) Standard 26262, or the British adoption of European Standard 50128.

NUREG-0800, Appendix 7.1-D, discusses software tools in the context of QA and reiterates the IEEE 7-4.3.2-2003 requirement that if defects not detected by software tools or introduced by software tools will not be detected by V&V activities, then the tool itself should be designed as safety-related software. All of the documents referenced by Appendix 7.1-D have been superseded so the guidance is dated.

NUREG/CR-6263 and NUREG/CR-6421 discuss vendor support of tools, configuration management (CM) of tools, the use of high-level languages with detailed programming standards, and a classification scheme for COTS software including software tools. NUREG/CR-6263 recommends that the system safety analysis should determine the extent of tool evaluations, an error analysis should determine the types of errors that can be introduced by the tool(s), and an external certification process should be used to certify the tool(s). NUREG/CR-6421 contains an acceptance process for COTS software in reactor operations that determines the acceptability of COTS software, including software tools, using a four level

classification scheme based on the importance to safety of the system in which the product will be used. The usefulness of both contractor reports are limited by their age and the reliance on standards that have been amended or superseded.

6.4 Institute of Electrical and Electronic Engineers

IEEE standards provide a few high-level software tool requirements and those standards are currently endorsed by NRC regulatory guides. The IEEE standards would not form a sound technical basis for developing new regulatory guidance for the commercial nuclear power industry because the standards lack detailed software tool verification, validation, QA, development, and dedication requirements. However, regulatory guidance for the commercial nuclear power industry could be improved by endorsing the latest standards and by influencing the IEEE to modify standards for consistency with other industry practices.

IEEE standards discuss software tools as an alternative to manual means to prepare specific documentation and perform V&V activities. IEEE standards also discuss concerns with common-cause failures and the selection of computer aided software engineering (CASE) tools. IEEE 1012-2012 discusses a four-level safety integrity level scheme for software and software tools which can be tailored to fit many applications; however, the requirements only apply to tools that insert or translate code and IEEE 7-4.3.2-2010, IEEE 7-4.3.2-2003, and NRC regulatory guides do not endorse the use of all four integrity levels.

IEEE 7-4.3.2-2010 and its predecessor, IEEE 7-4.3.2-2003, contain high-level requirements for software and software tool verification, validation, QA, development, and dedication. IEEE 7-4.3.2-2010 does not distinguish between the possible effects of using different types of tools and specifies the same requirements for all tools used in a development effort. However, the standard endorses qualifying a tool when it is developed within a high-quality life cycle process or by commercial dedication.

IEEE 14102-2010 and IEEE 1462-1998 give guidance on how to evaluate and select appropriate and reliable tools but does not address any method for qualifying them for use in developing safety critical software. IEEE 603-2009 gives guidance on designing safety systems to address common-cause failures. Although the standard does not specifically address software tools, a defect in a software tool used to develop software in multiple systems or used in multiple life cycle phases to develop software for a safety system has the potential of introducing a fault which could result in a common-cause failure. Tools should be sufficiently verified and validated when used in this manner to prevent the possibility of a common-cause failure.

6.5 International Electrotechnical Commission

IEC standards contain a generic probabilistic risk-based process for safety-related software development and a commercial nuclear power specific application of that generic process. IEC standards contains requirements and guidance for software tool selection, use, documentation, validation, qualification, and CM. IEC standards provide comprehensive software tool requirements and guidance that would form a sound technical basis for developing new regulatory guidance for the commercial nuclear power industry. However, the standards lack

consistency, lack a comprehensive review and approval process, and the probabilistic risk-based process to design safety-related systems may not be appropriate for commercial nuclear power safety-related software development since software failures are impossible to quantify.

IEC 61508 defines four SILs (i.e., SIL1 through SIL4) for classifying safety-related systems and software and defines three classes of software tools (i.e. T1, T2, and T3) for determining the rigor required for selecting, documenting, and validating the tools. However, the requirements are high-level and the concept of tool classes is not carried over to lower-level IEC standards (i.e., IEC 60880 and IEC 62138) to customize the tool qualification requirements based on the tool class.

IEC 61226 establishes a method of classification of instrumentation and control (I&C) systems and equipment according to their importance to safety. The classification of I&C functions (i.e., A, B, C, or non-classified) depends on their contribution to the prevention and mitigation of postulated initiating events. The resulting classification is then used in IEC 60880 and IEC 62138 to determine relevant specification and design requirements including qualification requirements for software tools used to develop I&C software and systems important to safety.

IEC 60880 addresses software aspects of computer-based systems performing category A functions as defined by IEC 61226 while IEC 62138 addresses software aspects of computer-based systems performing category B and C functions as defined by IEC 61226. IEC 60880 requires documentation of tool use, adequate training on tool use, and verification of tools to a level consistent with tool reliability requirements, the type of tool, and the potential of the tool to introduce faults. IEC 60880 and IEC 62138 identify types of tools that require different levels of verification and assessment. This treatment of tool types differs from the tool classes used in IEC 61508 which is the generic standard upon which IEC 60880 and IEC 62138 are based. The tool classes in IEC 61508 should be carried over to IEC 60880 and IEC 62138 for uniformity and consistency.

IEC 60880 attempts to tailor the qualification requirements for tools by requiring that tools be qualified considering the reliability requirements of the tool and the type of tool; however, IEC 60880 lacks specific details of the tailoring process. IEC 60880 specifies five tool qualification methods but does not recommend any specific combination of methods for the different tool types. IEC 60880 does not require tool qualification if the tool cannot introduce faults into the software, the tool output is always systematically verified, or tool faults are mitigated. IEC 60880 discourages the use of optimization when compiling the executable code.

IEC 62138 requirements for software tools are consistent with but less rigorous than the IEC 60880 requirements. IEC 62138 focuses efforts on tool certification and the certification of the tool development process rather than a rigorous tool qualification process.

6.6 Automobile Industry

ISO 26262 contains an automotive specific application of the generic IEC 61508 safety-related system development process. ISO 26262 contains requirements and guidance for software tool selection, use, documentation, and qualification. ISO 26262 provides comprehensive, well-

organized software tool requirements and guidance that would form a sound technical basis for developing new regulatory guidance for the commercial nuclear power industry. However, the standards lack a comprehensive review and approval process and the probabilistic risk-based process for designing road vehicle safety-related systems may not be appropriate for commercial nuclear power safety-related software development since software failures are impossible to quantify.

ISO 26262 uses four automotive safety integrity levels (ASIL A through D) to tailor the requirements of ISO 26262 to avoid unreasonable residual risk. ISO 26262 requires software to be assigned to one of four ASIL categories on the basis of the system ASIL and the level of risk associated with the use of the software in the system.

ISO 26262 contains a few high-level requirements for planning, selecting, using, and documenting software tools and for selecting a modeling or programming language. ISO 26262 lacks requirements for software tool training and CM. ISO 26262 permits the use of assembly language in road vehicle safety-related software that should not be allowed in commercial nuclear power safety-related system software based on the additional complexity of code inspections, bi-directional tracing of code to requirements, and the specialized skill set needed to write and perform QA on assembly language code.

ISO 26262 requires that a tool confidence level (TCL) be determined when a software tool supports or enables the tailoring of activities and tasks of the safety life cycle and the output of the software tool has not been examined or verified. This practice is consistent with the civil aviation industry as described in RTCA DO-178C; however, the ISO 26262 TCL process does not distinguish between software development tools and verification tools and the TCL is based on a probabilistic assessment that a malfunction and its corresponding erroneous output will be prevented or detected. The TCL is used along with the ASIL to select an appropriate combination of four methods of qualifying software tools. The four qualification methods are: confidence from prior tool use, evaluation of the tool development process, tool validation, and tool development in accordance with a safety standard. Although ISO 26262 identifies four alternative software tool qualification methods, the requirements for software tool qualification are not specific enough to ensure a consistent methodology within the automotive industry. Combining the best practices from the automotive industry with the best practices from the civil aviation industry would be the best technical basis upon which to improve existing regulatory guidance or develop new regulatory guidance for the commercial nuclear power industry.

6.7 Railway Industry

The railway industry uses a probabilistic risk-based process for safety-related system and software development. BS EN 50128 contains requirements and guidance for software tool selection, use, documentation, and qualification. BS EN 50128 also provides a comprehensive process for determining the confidence level in software tools used in the software development process similar to the tool qualification level used in the civil aviation industry. BS EN 50128 provides comprehensive, well-organized software tool requirements and guidance that would form a sound technical basis for developing new regulatory guidance for the commercial nuclear power industry. However, the standard lacks a comprehensive review and approval process,

and the probabilistic risk-based process for designing railway safety-related systems may not be appropriate for commercial nuclear power safety-related software development since software failures are impossible to quantify.

BS EN 50128 is a railway-specific implementation of IEC 61508 and specifies technical requirements for the development of safety-related software for railway control and protection systems. This standard requires a systematic approach for safety-related software development to identify hazards, assess risks, and identify risk reduction measures. BS EN 50128 requires software to be assigned to one of five SILs from SIL0 (lowest) to SIL5 (highest) on the basis of the system SIL and the level of risk associated with the use of the software in the system. However, only three SILs are actually used and the safety-related software SIL and the tables of recommendations for measures and techniques to satisfy the requirements of BS EN 50128 do not impact the selection, use, or qualification of software tools used in the safety-related software development processes.

BS EN 50128 contains several requirements for software tools used in the safety-related software development life cycle process that only depend on the tool functionality. Tools are assigned to one of three classes (i.e., Class T1, T2, or T3) and the tool requirements vary by class. An ISL observation, based on an analysis of BS EN 50128 tool requirements, is that some of the class T1 tools (i.e., design support tools and requirements specification tools) should be subject to more requirements including validation and verification since errors in these tools can ultimately impact the safety-related software design and functionality. Further, the requirements for class T2 tools should be expanded to include V&V and the requirements for class T3 tools should be expanded to require project-specific application evaluation of tools rather than relying on a high-level tool evaluation for an entire class of applications and a check of general suitability for the specific application.

6.8 International Atomic Energy Agency

IAEA safety guide IAEA NS-G-1.1 and draft safety guide IAEA DS-431 provide a few high-level software tool guidelines. Some of the guidance in these safety guides would form a good technical basis for developing new regulatory guidance for the commercial nuclear power industry because the guidance covers aspects of tool use that are not covered by other industries. However, the safety guides lack detailed software tool verification, validation, QA, development, and dedication guidance.

IAEA NS-G-1.1 is similar to BTP 7-14 in that it focuses on providing guidance for preparing documentation in each software development life cycle phase to demonstrate the safety of software used in nuclear power plant computer-based systems. IAEA NS-G-1.1 contains all the high-level, common industry requirements for software tool identification, configuration control, and documentation.

IAEA NS-G-1.1 does not define software classifications but does define two tool categories (i.e., software development tools and software verification tools) and provides tool requirements for each of those categories. Although IAEA NS-G-1.1 requires tool qualification and certification of the tool output, detailed tool qualification and certification processes, activities, and tasks

comparable to RTCA DO-178C and RTCA DO-330 are not provided. An ISL observation, based on an analysis of IAEA NS-G-1.1, is that tool categories should be expanded to cover all types of tools beyond development and verification tools and provide detailed qualification and certification requirements.

IAEA DS-431 is a new draft safety guide that will supersede IAEA NS-G-1.1 and IAEA NS-G-1.3 and accounts for the latest developments in I&C systems. IAEA DS-431 gives high-level guidance on the overall I&C architecture and on the I&C systems important to safety in nuclear power plants for the satisfaction of the safety goals of the plant. IAEA DS-431 maintains the similarity to BTP 7-14 by focusing on providing guidance for preparing documentation in each software development life cycle phase to demonstrate the safety of software important to safety used in nuclear power plant computer-based systems.

IAEA DS-431 contains all the high-level, common industry requirements for software tool identification, configuration control, and documentation. IAEA DS-431 eliminates the two software tool categories of development and verification tools that are specified in IAEA NS-G-1.1 in favor of providing examples of software tools that are used throughout the software development life cycle. IAEA DS-431 requires tools to be verified and assessed consistent with tool reliability requirements, the type of tool, the potential of the tool to introduce fault or fail to make the user aware of existing faults, and the extent to which the tool may affect redundant elements of a system or diverse systems. Tools that can introduce faults or fail to detect faults need to be verified to a greater extent than other tools; however, verification is not necessary if the tool output is systematically and independently verified. However, IAEA DS-431 does not supply any details on verification methods or assessment techniques. The high-level guidance of IAEA DS-431 needs to be supplemented with low-level requirements which tailor the requirements to the different types of tools.

6.9 Electric Power Research Institute

EPRI documents focus on commercial-grade dedication of hardware and software used in nuclear safety-related applications. The commercial-grade dedication process discussed in these EPRI documents should form a sound technical basis for developing new software tool regulatory guidance when supplemented with guidance from other industries.

Although the reports focus on hardware and software, the commercial-grade dedication process described in these reports can be applied to software tools used in the development of safety system software. An ISL observation, based on an analysis of the commercial-grade dedication process is that although the commercial-grade dedication process can be applied to software tools, many of the steps described in the dedication process, that are specific to hardware or software dedication, would be difficult to accomplish and must be adapted for software tools. Also, a software tool developed in accordance with a formal and recognized software life cycle development process is not considered a CGI and cannot be dedicated based on current rules. Therefore, the commercial-grade dedication process can only be applied to COTS software tools. Software tools developed under a 10 CFR Part 50, Appendix B, QA program can be safely used; however, another method of guaranteeing the safe use of software tools developed using a different QA process should be considered.

EPRI NP-5652 and EPRI TR-102260 contain a generic process and supplemental guidance for the acceptance of commercial-grade hardware components and parts for use in a nuclear safety-related application. The generic process consists of six steps: determining the safety function, confirming an item is a CGI, identifying critical characteristics, accepting the CGI, delegating the acceptance process within a 10 CFR Part 50, Appendix B, QA program, and controlling the reportability and traceability of the CGI. The most important part of the generic process is accepting the CGI. There are four acceptance methods: special tests and inspections, commercial-grade survey of the supplier, source verification, and acceptable supplier or item performance record.

Supplemental guidance in EPRI TR-102260 for the commercial-grade dedication of hardware components and parts includes the application of a graded approach to quality assurance to an extent consistent with the item's importance to safety. Based on guidance in EPRI NP-5652, the safety classification of a component is determined from an analysis of its safety function and is categorized as either safety-related or nonsafety-related. A graded approach would still be consistent with EPRI NP-5652 in that all safety-related components would need to be dedicated; however, the graded approach would subdivide the classification of safety-related into multiple classifications where the most important safety-related items are subjected to a more rigorous dedication process than less important safety-related items.

EPRI TR-106439 and EPRI TR-107339 extend the commercial-grade dedication process to commercial-grade digital equipment. EPRI TR-106439 provides guidance on the evaluation and acceptance of commercial-grade digital equipment and states that the key elements of the dedication process are a technical evaluation to define the requirements for the device, generating a set of critical characteristics for acceptance from the device requirements, and applying the methods of EPRI NP-5652 to verify the critical characteristics. Since EPRI NP-5652 applies primarily to hardware, EPRI TR-106439 states that new critical characteristics and additional verification activities are needed when the methods of EPRI NP-5652 are applied to digital devices. EPRI TR-106439 also states that no one acceptance method from EPRI NP-5652 will suffice by itself and that for many digital devices, a combination of acceptance methods are needed.

EPRI TR-107339 provides supplemental guidance for the technical evaluation and acceptance processes required for digital, software-based equipment and systems. The primary focus of EPRI TR-107339 is on the differences in the technical evaluation and acceptance process required for digital, software-based equipment and systems. EPRI TR-107339 provides guidance for each of the phases of the CGI dedication process including project definition, defining the detailed requirements, defining the critical characteristics for acceptance, formulation of an acceptance strategy, and verification of critical characteristics.

EPRI TR-1025243 provides guidance for the safety classification and acceptance of commercial-grade design and analysis tools used in nuclear safety-related applications via the commercial-grade dedication process. EPRI TR-1025243 provides a method for determining the functional safety classification of design and analysis tools that are not resident or installed as part of plant SSCs. The safety classification is determined on a case-by-case basis by

evaluating the actual functions of the computer program and its intended end uses. EPRI TR-1025243 identifies two deterministic options for classifying computer programs. One option considers failure modes and effects. The second option considers the impact of the computer program on SSCs. EPRI TR-1025243 also provides guidance for using a commercial-grade dedication methodology to accept safety-related, commercial-grade design and analysis tools used in the design and analysis of safety-related plant SSCs. EPRI TR-1025243 extends the commercial-grade dedication process for hardware components described in EPRI NP-5652 to design and analysis software tools used to support the hardware design and development. EPRI TR-1025243 guidance does not apply to all software tools used to support the design and development of safety-related DI&C software. However, with minor modifications to the safety classification and acceptance methodologies, the guidance could be extended to all software design and development tools.

As stated above, a software tool developed in accordance with a formal and recognized software life cycle development process is not considered a CGI and cannot be dedicated based on current rules. Therefore, the commercial-grade dedication process can only be applied to COTS software tools. The commercial-grade dedication of COTS software tools can be performed by initially conducting a safety evaluation to determine the safety classification of the tool, determining the COTS software tool critical characteristics, and then accepting the COTS software tool. The software tool is classified safety-related or nonsafety-related by an analysis of its safety function. Only safety-related software tools need to be dedicated. However, even safety-related software tools do not have to be dedicated if the tool output is independently verified each time it is used. The critical characteristics of the COTS software tool then have to be defined. An ISL observation, based on an analysis, is that tool qualification should be designated as a critical characteristic of a safety-related COTS software tool to support the acceptance process and that the COTS tool qualification processes of RTCA DO-330 should be considered in any new regulatory guidance. The software tool must then be accepted to complete the dedication process using one of four acceptance methods: special tests and inspections, commercial-grade survey of the supplier, source verification, and acceptable supplier or item performance record. These methods would have to be adapted to specifically address COTS software tools.

6.10 National Institute of Standards and Technology

National Institute of Standards and Technology (NIST) special publication (SP) 500-234 contains only a few high-level software tool requirements that deal with V&V and training associated with the health care industry. These high-level requirements are similar to requirements in IEEE 1012-2012; therefore, NIST SP 500-234 would be of little benefit as a technical basis for developing new regulatory guidance for the commercial nuclear power industry. Both NIST SP 500-234 and IEEE 1012-2012 recommend that an independent V&V (IV&V) team execute qualification tests on tools (e.g., compilers, assemblers, and utilities) shared with the development environment and recommend that the IV&V team should use or develop its own set of test and analysis tools when possible. Both NIST SP 500-234 and IEEE 1012-2012 also state that the V&V plan should include details for acquiring tools and for training personnel. However, NIST SP 500-234 does not discuss software integrity levels or provide

minimum V&V activities as a function of integrity level. NIST SP 500-234 does not distinguish between different categories of tools or use safety as a measure to determine tool qualification requirements. NIST SP 500-234 also does not provide guidance for developing the required independent software tools or provide a process for qualifying software tools. NIST SP 500-234 does discuss a need for artificial intelligence techniques and knowledge-based systems to manage the vast knowledge in the health care industry. However, these advanced computer based systems have no current application in the commercial nuclear power industry.

6.11 Atomic Energy of Canada, Ltd

Atomic Energy of Canada Limited (AECL) books, journals, standards, and technical reports contain only a few high-level software tool requirements and would be of little benefit as a technical basis for developing new regulatory guidance for the commercial nuclear power industry because all the high-level requirements are covered by other industries. AECL CE-1001-STD specifies requirements and a minimum set of software engineering processes for safety-critical software development for Canadian-invented Deuterium-Uranium (CANDU®) nuclear generating stations. Software tool requirements are limited to high-level requirements to identify necessary resources for tool development, identify and document tools in plans and procedures, identify personnel required to maintain and manage the tools; and identify the qualification requirements for software tools. AECL CE-1001-STD does not distinguish between different types of software tools, does not use software tool categories or classifications, and does not contain any specific software tool qualification requirements tailored to the type of tool.

7 References

7.1 Aerospace Industry

NPR 7150.2A	<i>"NASA Software Engineering Requirements"</i> , November 2009
NASA-STD-7009	<i>"Standard for Models and Simulations"</i> , July 2008
NASA-STD-8739.8	<i>"Software Assurance Standard"</i> , Change 1, July 2004
NASA-STD-8719.13B	<i>"Software Safety Standard"</i> , Change 1, July 2004
NASA-STD-8719.13C	<i>"Software Safety Standard"</i> , May 2013
NASA-GB-8719.13	<i>"NASA Software Safety Guidebook"</i> , March 2004
NASA-HDBK-8739.23	<i>"NASA Complex Electronics Handbook for Assurance Professionals"</i> , Change 1, March 2011

7.2 Civil Aviation Industry

FAA-STD-026A	<i>"Software Development for the National Airspace System (NAS)"</i> , June 2001
FAA Order 8110.49	<i>"Software Approval Guidelines"</i> , June 2003
FAA Order 8110.49	<i>"Software Approval Guidelines"</i> , Change 1, September 2011
FAA AC 20-148	<i>"Reusable Software Components"</i> , December 2004
FAA AC 20-115C	<i>"Airborne Software Assurance"</i> , July 2013
FAA Notice 8110.110	<i>"Software Approval Guidelines"</i> , January 2010
Position Paper CAST-13	<i>"Automatic Code Generation Tools Development Assurance"</i> , June 2002
Position Paper CAST-22	<i>"Reuse of Software Tool Qualification Data Across Company Boundaries (Applying the Reusable Software Component Concept to Tools)"</i> , Revision 1, March 2005
DOT/FAA/AR-06/54	<i>"Software Verification Tools Assessment Study"</i> , June 2007

7.3 Radio Technical Commission for Aeronautics

RTCA DO-178B	<i>"Software Considerations in Airborne Systems and Equipment Certification"</i> , December 1992
RTCA DO-178C	<i>"Software Considerations in Airborne Systems and Equipment Certification"</i> , December 2011
RTCA DO-248B	<i>"Final Report for Considerations of DO-178B 'Software Considerations in Airborne Systems and Equipment Certification'"</i> , October 2001
RTCA DO-248C	<i>"Supporting Information for DO-178C and DO-278^a"</i> , December 2011

RTCA DO-330 *“Software Tool Qualification Considerations”*, December 2011

7.4 Nuclear Regulatory Commission

Regulatory Guide 1.152 *“Criteria for Use of Computers in Safety Systems of Nuclear Power Plants”*, Revision 3, July 2011

Regulatory Guide 1.168 *“Verification, Validation, Reviews, and Audits for Digital Computer Software Used in Safety Systems of Nuclear Power Plants”*, Revision 2, July 2013

Regulatory Guide 1.169 *“Configuration Management Plans for Digital Computer Software Used in Safety Systems of Nuclear Power Plants”*, Revision 1, July 2013

Regulatory Guide 1.170 *“Test Documentation for Digital Computer Software Used in Safety Systems of Nuclear Power Plants”*, Revision 1, July 2013

Regulatory Guide 1.171 *“Software Unit Testing for Digital Computer Software Used in Safety Systems of Nuclear Power Plants”*, Revision 1, July 2013

Regulatory Guide 1.172 *“Software Requirements Specifications for Digital Computer Software and Complex Electronics Used in Safety Systems of Nuclear Power Plants”*, Revision 1, July 2013

Regulatory Guide 1.173 *“Developing Software Life Cycle Processes for Digital Computer Software Used in Safety Systems of Nuclear Power Plants”*, Revision 1, July 2013

Regulatory Guide 1.97 *“Instrumentation for Light-Water-Cooled Nuclear Power Plants to Assess Plant and Environs Conditions During and Following an Accident”*, Revision 3, May 1983

Regulatory Guide 1.97 *“Criteria for Accident Monitoring Instrumentation for Nuclear Power Plants”*, Revision 4, June 2006

Digital I&C-ISG-06 *“Digital Instrumentation and Controls Licensing Process”*, Revision 1, January 2011

NUREG-0800 BTP 7-14 *“Guidance on Software Reviews for Digital Computer-Based Instrumentation and Control Systems”*, Revision 5, March 2007

NUREG-0800 App 7.1-D *“Guidance for Evaluation of the Application of IEEE STD 7-4.3.2”*, March 2007

NUREG/CR-6101 J. Dennis Lawrence, *“Software Reliability and Safety in Nuclear Reactor Protection Systems”*, Lawrence Livermore National Laboratory, UCRL-ID-114839, June 1993

NUREG/CR-6263 S. Seth, W. Bail, D. Cleaves, H. Cohen, D. Hybertson, C. Schaefer, G. Stark, A. Ta, B. Ulery, *“High Integrity Software for Nuclear Power Plants: Candidate Guidelines,*

	<i>Technical Basis, and Research Needs</i> ”, The Mitre Corporation, MTR 94W0000114, Vol. 2, June 1995
NUREG/CR-6421	G. G. Preckshot, J. A. Scott, “ <i>A Proposed Acceptance Process for Commercial Off-the-Shelf (COTS) Software in Reactor Applications</i> ”, Lawrence Livermore National Laboratory, UCRL-ID-122526, March 1996
Generic Letter 89-02	“ <i>Actions to Improve the Detection of Counterfeit and Fraudulently Marketed Products</i> ”, March 1989
Generic Letter 91-05	“ <i>Licensee Commercial-Grade Procurement and Dedication Programs</i> ”, April 1991

7.5 National Standards

IEEE 7-4.3.2-2003	“ <i>IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations</i> ”, December 2003
IEEE 7-4.3.2-2010	“ <i>IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations</i> ”, August 2010
IEEE 7-4.3.2-20XX	“ <i>IEEE Standard Criteria for Programmable Digital Devices in Safety Systems of Nuclear Power Generating Stations</i> ”, August 2010
IEEE 100-2000	“ <i>IEEE 100 The Authoritative Dictionary of IEEE Standards Terms</i> ”, Seventh Edition, February 2007
IEEE 279-1971	“ <i>IEEE Standard: Criteria for Protection Systems for Nuclear Power Generating Stations</i> ”, 1971
IEEE 603-1991	“ <i>IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations</i> ”, 1991
IEEE 603-1998	“ <i>IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations</i> ”, 1998
IEEE 603-2009	“ <i>IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations</i> ”, November 2009
IEEE 730-2002	“ <i>IEEE Standard for Software Quality Assurance Plans</i> ”, 2002
IEEE 730-2014	“ <i>IEEE Standard for Software Quality Assurance Processes</i> ”, June 2014
IEEE 828-2005	“ <i>IEEE Standard for Software Configuration Management Plans</i> ”, August 2005
IEEE 828-2012	“ <i>IEEE Standard for Configuration Management in Systems and Software Engineering</i> ”, March 2012
IEEE 829-2008	“ <i>IEEE Standard for Software and System Test Documentation</i> ”, July 2008

IEEE 830-1998	<i>"IEEE Recommended Practice for Software Requirements Specifications", 1998</i>
ANSI/IEEE 1008-1987	<i>"IEEE Standard for Software Unit Testing", 1986, Reaffirmed 1993</i>
IEEE 1012-1998	<i>"IEEE Standard for Software Verification and Validation", July 1998</i>
IEEE 1012-2004	<i>"IEEE Standard for Software Verification and Validation", June 2005</i>
IEEE 1012-2012	<i>"IEEE Standard for System and Software Verification and Validation", May 2012</i>
IEEE 1028-2008	<i>"IEEE Standard for Software Reviews and Audits", August 2008</i>
ANSI/IEEE 1042-1987	<i>"IEEE Guide to Software Configuration Management", September 1988, Reaffirmed 1993</i>
IEEE 1074-1991	<i>"IEEE Standard for Developing Software Life Cycle Processes", 1992</i>
IEEE 1074-1995	<i>"IEEE Standard for Developing Software Life Cycle Processes", 1996</i>
IEEE 1074-2006	<i>"IEEE Standard for Developing a Software Project Life Cycle Process", July 2006</i>
IEEE 1462-1998	<i>"Information Technology - Guideline for the Evaluation and Selection of CASE Tools", March 1998, Reaffirmed December 2004</i>
IEEE/EIA 12207.0-1996	<i>"Industry Implementation of International Standard ISO/IEC 12207:1995, (ISO/IEC 12207) Standard for Information Technology - Software Life Cycle Processes", March 1998</i>
IEEE/EIA 12207.1-1997	<i>"Guide for Information Technology - Software Life Cycle Processes - Life Cycle Data", April 1998</i>
IEEE/EIA 12207.2-1997	<i>"Industry Implementation of International Standard ISO/IEC 12207:1995, (ISO/IEC 12207) Standard for Information Technology - Software Life Cycle Processes - Implementation Considerations", 1998</i>
IEEE 12207-2008	<i>"Systems and Software Engineering - Software Life Cycle Processes", Second Edition, February 2008</i>
IEEE 14102-2010	<i>"IEEE Standard for Adoption of ISO/IEC 14102:2008, Information Technology - Guideline for the Evaluation and Selection of CASE Tools", September 2010</i>
IEEE 15288-2008	<i>"Systems and Software Engineering - System Life Cycle Processes", Second Edition, February 2008</i>
IEEE 24765-2010	<i>"Systems and Software Engineering – Vocabulary", December 2010</i>

IEEE 29148-2011 “*Systems and Software Engineering - Life Cycle Processes - Requirements Engineering*”, First Edition, December 2011

7.6 International Standards

IEC 60880 Ed. 1.0 “*Software for Computers in the Safety Systems of Nuclear Power Stations*”, First Edition, September 1986

IEC 60880-2 Ed 1.0 “*Software for Computers Important to Safety for Nuclear Power Plants – Part 2: Software Aspects of Defense Against Common Cause Failures, Use of Software Tools and of Pre-Developed Software*”, First Edition, December 2000

IEC 60880 Ed. 2.0 “*Nuclear Power Plants - Instrumentation and Control Systems Important to Safety - Software Aspects for Computer-Based Systems Performing Category A Functions*”, Second Edition, May 2006

IEC 60987 Ed. 2.0 “*Nuclear Power Plants - Instrumentation and Control Important to Safety – Hardware Design Requirements for Computer-Based Systems*”, Second Edition, August 2007

IEC 61226 Ed. 3.0 “*Nuclear Power Plants – Instrumentation and Control Important to Safety – Classification of Instrumentation and Control Functions*”, Third Edition, July 2009

IEC 61508 Ed. 2.0 “*Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*”, Second Edition, April 2010

IEC 61508-0 Ed. 1.0 “*Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems - Part 0: Functional Safety and IEC 61508*”, First Edition, January 2005

IEC 61508-1 Ed. 2.0 “*Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems - Part 1: General Requirements*”, Second Edition, April 2010

IEC 61508-2 Ed. 2.0 “*Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems - Part 2: Requirements for Electrical/Electronic/Programmable Electronic Safety-Related Systems*”, Second Edition, April 2010

IEC 61508-3 Ed. 2.0 “*Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems - Part 3: Software Requirements*”, Second Edition, April 2010

IEC 61508-4 Ed. 2.0 “*Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems - Part 4: Definitions and Abbreviations*”, Second Edition, April 2010

IEC 61508-5 Ed. 2.0 “*Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems - Part 5: Examples of Methods for the Determination of Safety Integrity Levels*”, Second Edition, April 2010

IEC 61508-6 Ed. 2.0	<i>“Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems - Part 6: Guidelines on the Application of IEC 61508-2 and IEC 61508-3”, Second Edition, April 2010</i>
IEC 61508-7 Ed. 2.0	<i>“Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems - Part 7: Overview of Techniques and Measures”, Second Edition, April 2010</i>
IEC 61513 Ed. 2.0	<i>“Nuclear Power Plants – Instrumentation and Control Important to Safety – General Requirements for Systems”, Second Edition, August 2011</i>
IEC 62138 Ed. 1.0	<i>“Nuclear Power Plants - Instrumentation and Control Important for Safety - Software Aspects for Computer-Based Systems Performing Category B or C Functions”, First Edition, January 2004</i>
IEC 62340 Ed. 1.0	<i>“Nuclear Power Plants - Instrumentation and Control Systems Important to Safety - Requirements for Coping with Common Cause Failure (CCF)”, First Edition, December 2007</i>
ISO/IEC 12207:1995	<i>“Information Technology – Software Life Cycle Processes”, 1995</i>
ISO/IEC 14102:1995	<i>“Information Technology – Guideline for the Evaluation and Selection of CASE Tools”, 1995</i>
ISO 26262-1:2011	<i>“Road Vehicles – Functional Safety – Part 1: Vocabulary”, November 2011</i>
ISO 26262-2:2011	<i>“Road Vehicles – Functional Safety – Part 2: Management of Functional Safety”, November 2011</i>
ISO 26262-3:2011	<i>“Road Vehicles – Functional Safety – Part 3: Concept Phase”, November 2011</i>
ISO 26262-4:2011	<i>“Road Vehicles – Functional Safety – Part 4: Product Development at the System Level”, November 2011</i>
ISO 26262-5:2011	<i>“Road Vehicles – Functional Safety – Part 5: Product Development at the Hardware Level”, November 2011</i>
ISO 26262-6:2011	<i>“Road Vehicles – Functional Safety – Part 6: Product Development at the Software Level”, November 2011</i>
ISO 26262-7:2011	<i>“Road Vehicles – Functional Safety – Part 7: Production and Operation”, November 2011</i>
ISO 26262-8:2011	<i>“Road Vehicles – Functional Safety – Part 8: Supporting Processes”, November 2011</i>
ISO 26262-9:2011	<i>“Road Vehicles – Functional Safety – Part 9: Automotive Safety Integrity Level (ASIL)-Oriented and Safety-Oriented Analyses”, November 2011</i>
ISO 26262-10:2012	<i>“Road Vehicles – Functional Safety – Part 10: Guideline on ISO 26262”, July 2012</i>

BS EN 50128:2011	<i>“Railway Applications – Communication, Signalling and Processing Systems – Software for Railway Control and Protection Systems”</i> , British Standards Institution, British-Adopted European Standard, July 2011
BS EN 50126-1:1999	<i>“Railway Applications – The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS) – Part 1: Basic Requirements and Generic Process”</i> , British Standards Institution, British-Adopted European Standard, December 1999
BS EN 50129:2003	<i>“Railway Applications – Communication, Signalling and Processing Systems – Safety Related Electronic Systems for Signalling”</i> , British Standards Institution, British-Adopted European Standard, May 2003
IAEA DS-431	<i>“Design of Instrumentation and Control Systems for Nuclear Power Plants”</i> , Draft Safety Guide, July 2013
IAEA NS-R-1	<i>“Safety of Nuclear Power Plants: Design - Safety Requirements”</i> , 2000
IAEA NS-G-1.1	<i>“Software for Computer Based Systems Important to Safety in Nuclear Power Plants – Safety Guide”</i> , 2000
IAEA NS-G-1.3	<i>“Instrumentation and Control Systems Important to Safety in Nuclear Power Plants - Safety Guide”</i> , 2002
IAEA SSG-30	<i>“Safety Classification of Structures, Systems, and Components in Nuclear Power Plants – Specific Safety Guide”</i> , 2014
IAEA SSR-2/1	<i>“Safety of Nuclear Power Plants: Design – Specific Safety Requirements”</i> , 2012
AECL CE-0100-STD	<i>“Standard for Computer System Engineering”</i> , CANDU Computer Systems Engineering Centre of Excellence, Revision 0, December 1999
AECL CE-0101-STD	<i>“Standard for Software Engineering of Category II Software and Category III Software”</i> , CANDU Computer Systems Engineering Centre of Excellence, Revision 0, 2000
AECL CE-1001-STD	<i>“Standard for Software Engineering of Safety Critical Software”</i> , CANDU Computer Systems Engineering Centre of Excellence, Revision 2, December 1999

7.7 Electric Power Research Institute

EPRI NP-5652	<i>“Guideline for the Utilization of Commercial Grade Items in Nuclear Safety Related Applications (NCIG-07)”</i> , June 1988
EPRI TR-102260	<i>“Supplemental Guidance for the Application of EPRI Report NP-5652 on the Utilization of Commercial Grade Items”</i> , March 1994

EPRI TR-106439	<i>“Guideline on Evaluation and Acceptance of Commercial Grade Digital Equipment for Nuclear Safety Applications”</i> , October 1996
EPRI TR-107330	<i>“Generic Requirements Specification for Qualifying a Commercially Available PLC for Safety-Related Applications in Nuclear Power Plants”</i> , December 1996
EPRI TR-107339	<i>“Evaluating Commercial Digital Equipment for High Integrity Applications”</i> , December 1997
EPRI TR-108831	<i>“Requirements Engineering for Digital Upgrades: Specification, Analysis, and Tracking”</i> , December 1997
EPRI TR 1025243	<i>“Plant Engineering: Guideline for the Acceptance of Commercial-Grade Design and Analysis Computer Programs Used in Nuclear Safety-Related Applications”</i> , Revision 1, December 2013

7.8 National Institute of Standards

NIST SP 500-234	<i>“Reference Information for the Software Verification and Validation Process”</i> , April 1996
-----------------	--

7.9 Miscellaneous

ASME NQA-2a-1990, *“Quality Assurance Requirements of Computer Software for Nuclear Facility Operations”*, 1990

B. Selic, *“The Pragmatics of Model-Driven Development”*, IEEE Computer Society, IEEE Software, Volume 20, Issue 5, 2003

Elsevier Journal, *“Mechatronics, The Science of Intelligent Machines”*, International Federation of Automatic Control, Volume 24, Issue 3, April 2014

Health and Safety Executive in the United Kingdom, *“IEC 61508 Overview Report”*, Version 2.0, January 2006

J. Servatius, P. Butchart, *“Evaluation of Guidance for Tools Used to Develop Safety-Related Digital Instrumentation and Control Software for Nuclear Power Plants – Task 1 Report: Survey of the State of Practice”*, Information Systems Laboratories Inc., ISL-ESD-TR-13-05, July 2013

L. Zuck, *“A Methodology for the Translation Validation of Optimizing Compilers”*, Journal of Universal Computer Science, Volume 9, No. 3, 2003

MDEP Generic Common Position DICWG No 2, *“MDEP Common Position on Software Tools for the Development of Software for Safety Systems”*, Multinational Design Evaluation Programme, Digital I&C Working Group, December 2010

M. Hall, D. Padua, K. Pingalli, *“Compiler Research: The Next 50 Years”*, Communications of the ACM, Volume 52, No. 2, 2009

NITSL-SQA-2005-02, "*Guidance Document to Implement Policy for Software Quality Assurance in the Nuclear Power Industry*", National Energy Institute, Revision 1, January 2009

Oracle Corporation, "*Oracle Binary Code License Agreement for the Java SE Platform Products and JavaFX*", April 2013

Intentionally Blank