

**APPENDIX A**

**SUMMARY REPORT OF THE  
EXTERNAL REVIEW PANEL MEETING ON  
RELIABILITY MODELING OF DIGITAL SYSTEMS  
(MAY 23-24, 2007)**

# TABLE OF CONTENTS

	<u>Page</u>
A.1. INTRODUCTION .....	A-1
A.2. PRESENTATION OF PRELIMINARY COMMENTS BY EACH PANEL MEMBER .....	A-3
A.3. TRADITIONAL METHODS AND THEIR APPLICATIONS .....	A-13
A.3.1 Defining and Identifying “Traditional” Methods .....	A-13
A.3.2 Conclusions Stated versus Applications of the Criteria Presented in the Report .....	A-15
A.3.3 Criteria Applied to Methods or to Models/Applications .....	A-15
A.4. COMMENTS ON REVIEW CRITERIA .....	A-17
A.4.1 General Comments on Review Criteria .....	A-17
A.4.2 Category 1: Level of Detail of the Model .....	A-17
A.4.3 Category 2: Identification of Failure Modes of the Components of Digital Systems .....	A-18
A.4.4 Category 3: Software Failures .....	A-19
A.4.5 Category 4: Modeling of Dependencies .....	A-20
A.4.5.1 Subcategory 4.1: Communication Networks/Buses .	A-21
A.4.5.2 Subcategory 4.3: Support Systems .....	A-21
A.4.5.3 Subcategory 4.4: Sharing Hardware .....	A-22
A.4.5.4 Subcategory 4.5: Interactions of Digital Systems with Other Systems .....	A-22
A.4.5.5 Subcategory 4.6: Modeling of Fault Tolerance Features .....	A-23
A.4.5.6 Subcategory 4.2: Common Cause Failures .....	A-24
A.4.6 Category 7: Probabilistic Data .....	A-24
A.4.7 Criteria 8.4 and 8.5: Uncertainty .....	A-25
A.4.8 Category 6: Ease of Integration with a PRA Model .....	A-26
A.4.9 Category 5: Human Errors .....	A-27
A.4.10 Criteria 8.1 - 8.3: Documentation and Results .....	A-27
A.5. CONCLUDING REMARKS .....	A-28
Attachment A	
Agenda of External Review Panel Meeting on Selection of Traditional Methods for Reliability Modeling of Digital Systems .....	A-30

Attachment B

List of Documents Sent to Panel Members ..... A-32

Attachment C

Expert Panel Meeting Attendees ..... A-33

Attachment D

Biographies of Panel Members ..... A-34

Attachment E

Written Comments Provided by Reviewer A ..... A-36

Attachment F

Written Comments Provided by Reviewer C ..... A-46

Attachment G

Written Comments Provided by Reviewer D ..... A-48

Attachment H

Order for Addressing Review Criteria Categories/Subcategories ..... A-52

## **A.1. INTRODUCTION**

This summary report documents the discussions that took place at an external review panel meeting on traditional methods for modeling digital systems as part of a probabilistic risk assessment (PRA). The meeting was held at Brookhaven National Laboratory (BNL) on May 23 and 24, 2007.

### **A.1.1 Background**

The U.S. Nuclear Regulatory Commission (NRC) is currently performing research on digital system risk assessment. Their objective is to identify and develop methods, analytical tools, and regulatory guidance to support (1) using risk information on digital systems in nuclear power plant (NPP) licensing decisions, and, (2) including models of digital systems into NPPs' PRAs.

The NRC is exploring, in parallel, both the dynamic and traditional methods of modeling digital-system reliability. For this research, the latter can be thought of as the more well-established and commonly used methods of NPP system reliability modeling (e.g., fault tree modeling). Dynamic methods can be thought of as methods of NPP system-reliability modeling that attempt to explicitly model the coupling between a digital system and the plant's physical processes. (Note: The distinction between traditional and dynamic methods was further discussed during the meeting, as documented in Section 3.1 of this report.)

Under a contract with the NRC, BNL conducted the following work as part of the research on traditional methods of modeling digital system reliability:

1. Identified for further exploration two traditional methods that represent a spectrum of capabilities for modeling and quantitatively assessing the reliability of digital systems.
2. Developed criteria for evaluating reliability models of digital systems, which could eventually provide input to the technical basis for risk-informed decision-making.
3. Reviewed reliability models developed using traditional methods, such as fault tree and Markov methods, against the criteria to assist in determining the capabilities and limitations of the state-of-the-art of traditional reliability models.

The findings of this work are documented in a draft letter report (T. L. Chu, G. Martinez-Guridi, M. Yue, and J. Lehner, "Probabilistic Modeling of Digital Systems at Nuclear Power Plants: Traditional Methods Selection," Brookhaven National Laboratory, Draft Letter Report, April 2007).

### **A.1.2 External Review Panel Meeting Process**

To more fully involve the technical community in identifying the most promising traditional methods for reliability modeling of digital systems, and in developing criteria for evaluating such reliability models, an external review panel was established to review the findings from the BNL activities. The panel was comprised of six members, all of whom have expertise in modeling and

quantifying digital-system reliability, as well as in PRA. The objectives of the external review panel were to assess the following:

1. The identification of traditional methods and their application.
2. The draft criteria used to review reliability models of digital systems.
3. The limitations of the state-of-the-art in modeling digital systems.

The responsibilities of the panel members were to (1) study BNL's draft letter report before the meeting of the expert panel, and (2) attend the two-day 1 meeting to satisfy the objectives listed above. Attachment A of this report contains the agenda for the meeting.

Panel members received a set of background information before the meeting. In addition setting out the objectives of the meeting and describing the process to be followed during the meeting, the background information contained the BNL draft letter report, and other related papers and reports. The materials sent to the panel members are listed in Attachment B. Furthermore, each member was asked to judge beforehand whether the draft criteria for evaluating reliability models of digital systems were appropriate and to identify any additions, deletions, or modifications to them.

The meeting took place at BNL on May 23 and 24, 2007. It was conducted with the help of a facilitator who was responsible for aiding the discussions and keeping them focused. In addition to the panel members and the facilitator, the meeting was attended by the authors of the BNL report and the NRC Project Manager. Their role was to provide answers and clarifications in response to the panel members' questions. Attachment C of this summary report list the attendees' names; brief biographies of the experts are given in Attachment D.

The remaining sections of this report summarize the discussions of the meeting roughly in chronological order. At the beginning of the meeting, each panel member gave a short presentation of his preliminary thoughts on the BNL report (documented in Section 2 of this report). The next major topic for discussion was the identification of traditional methods and their relevant applications, as summarized in Section 3. Section 4 of this report summarizes the discussions on the proposed criteria, in the same order that their categories were reviewed. The concluding remarks are summarized in Section 5.

It should be noted that it was not a goal of the meeting to obtain a consensus among the panel on any particular aspect of the work under review. Rather, the goal was to receive feedback from a broad spectrum of individuals who have significant experience in the subject. Accordingly, while the following sections report any points of general agreement among the panel members, most of the information is in the form of comments from individual panel members.

## **A.2. PRESENTATION OF PRELIMINARY COMMENTS BY EACH PANEL MEMBER**

During the morning of May 23, 2007, each member of the external review panel gave a short presentation of his preliminary thoughts on using traditional methods for reliability modeling of digital systems, such as suggestions for alternative methods/applications, and on the draft criteria used for reviewing the digital system reliability models.

Points of general agreement among the panel members include the following:

- The BNL report contains valuable information about traditional methods.
- A substantial amount of probabilistic data of digital components/systems has been generated, but most of it is not publically available. An important and difficult issue is how to obtain this data.
- The term “criterion” should be used instead of the term “requirement” in the report due to the regulatory implications of the word “requirement.”

The comments by individual members of the panel are presented next, in the order in which they gave their presentations.

### **Reviewer A<sup>1</sup>**

General comments relative to the approach to be used in the evaluation of modeling methods and their application to digital systems:

- The term “digital systems” is very broad and covers a whole spectrum of systems with very different characteristics, therefore:
  - Different modeling methods may be needed for different areas of application within this broad spectrum.
  - It may not be possible to apply the same general criteria for evaluation of methods that are intended to address a specific area of application rather than another. For example, by and large, one can see a digital system as comprising three basic layers of components: hardware, operating system, and application software, plus the “external balance-of-system” with which the digital system interface. Each layer has different a different type of functionality and is subject to different types of faults and failures. Thus different types of modeling and model evaluation criteria may apply to the different layers.

---

<sup>1</sup> In addition, to his comments during the panel meeting, reviewer A provided some comments after the meeting that clarify and expand his points of view. Accordingly, this Appendix includes all his comments.

- There is a big difference between evaluating the suitability of a modeling framework or method to cover a range of possible applications, and evaluating one specific application of the method, in light of a set of criteria that particular application of the method was not set out to satisfy in the first place. By adopting the latter mode of evaluation it is very difficult to understand and assess the true strength of a framework or method.

Observations on operating experience of mission-critical digital systems and NPP digital systems modeling needs:

- National Aeronautics and Space Administration (NASA) operational experience indicates that a majority of the mission failures that have occurred and in which digital control systems and associated software were involved occurred due to system and software design errors, which became failures in mission execution because the digital system and software had to face unanticipated system conditions. Failures due to other causes (e.g., software coding errors) have not occurred in mission-critical systems, probably because they can be identified and eliminated with traditional validation and verification techniques before mission execution.
- In addressing NPP digital system modeling needs, it is important to have the ability to adapt the modeling approach to the particular type of system and interactions that need to be modeled. It is not prudent to suggest, as often is suggested by the industry, that for risk assessment and safety purposes it is sufficient to address only those systems categorized as “safety related” (i.e., RPS and ESFAS), whereas “non-safety related” systems are of secondary importance. There are a number of good reasons why a flexible portfolio of modeling tools that covers both “safety related” and “non-safety related” systems:
  - In a NPP, serious challenges to operational safety may come from systems that are nominally categorized as “non-safety related.” For example, the Three-Mile Island accident occurred not because of safety related systems failures, but because of triggering events in the non-safety related feedwater system, and interactions between the plant systems and their human operators.
  - An RPS is based on open loop logic and in that respect can be probably tested satisfactorily using traditional validation and verification techniques. The same is not true of a digital feedwater and level control system, which has logic and timing-dependent control loops and is also potentially subject to the effect of human errors introduced by its interface with human operators.
  - In general, interconnected control systems are considerably more complex than safety systems (as defined in the NPP context), because of their combination of logic, algorithms, and human interfaces.
- In light of all the above, one can conclude that “traditional methods” may be perhaps adequate to deal with safety systems that are relatively simple (in their logic and degree of permitted interaction with other systems). However, more advanced dynamic modeling

methods appear to be definitely needed to address the potential of system failures initiated by control systems and unnecessary challenges to safety systems that may progress to unexpected and undesirable consequences.

Comments on the nature and suitability of some of the methods evaluated by BNL:

- Military Handbook 217 (MIL-HDBK-217) is hardly a method for modeling digital systems, even though it contains information of how to assess the reliability of certain hardware components of digital system. Since 217 has not been supported by the Government since the early 1990's, its information is also based on outdated data. The more recent evolution of 217, 217 Plus, is based on unverified data and is still confined to estimating the reliability of electronic hardware components.
- “Traditional Markov” models may be a good way of modeling fault interactions within a digital system, but it doesn't necessarily address in a satisfactory way the interactions between the functions of a digital system and the operational behavior of the “balance-of-system” and controlled equipment.

Comments on the objective of modeling and the establishment of criteria to judge the quality of modeling approaches and modeling results:

- Basic objectives of digital system risk modeling that appear to be realistically pursuable are:
  - Identification of significant digital system failure modes, with particular emphasis on those that are related to interactions between digital system software and controlled system functions, since these are often the most difficult to understand and uncover.
  - Identification of the type and degree of testing, explicitly including systematic software testing, that is needed to “bound” the level of risk contribution that can be expected from a particular digital system. This may not be as difficult to achieve as commonly believed to be, because software failures are usually triggered by the occurrence of specific system conditions, which in turn may occur with a certain frequency. The condition frequencies are equivalent to unconditional “hazard rate,” which can be assessed independently from the software test process, whereas the actual software failure probabilities are conditional probabilities that can be determined by testing the software within the input space defined by each system condition of interest. I.e., if an input condition is expected to occur with a frequency of  $10^{-3}$  per year, then it may be sufficient to “explore” the software input space defined by such a condition by means of random, but systematic, testing repeated a minimum of 1000 times without encountering a failure, in order to “bound” the risk level associated with the occurrence of that condition at an order of magnitude of  $10^{-6}$  per year.



- With respect to the development of good criteria for evaluation of approaches to digital systems failure and risk modeling, the following consideration should apply:
  - A fundamental distinction needs to be made between establishing criteria to judge the quality and effectiveness of a modeling method, and criteria to judge whether one specific application of a method meets certain specific objectives. For example: fault tree analysis is sometimes used – without cut set quantification – to aid a system failure investigation process. It would be erroneous to pick up one such fault tree analysis application and conclude that fault tree analysis, as a method in general, fails to meet a criterion requiring risk quantification.
  - Consistent with the above it would be appropriate and advisable for the BNL study to shift emphasis from evaluating specific past methodology applications to evaluating the suitability of specific aspects of a methodology to being applied effectively for the purpose of digital systems failure and risk modeling, according to foreseeable NRC regulatory evaluation needs.
  - Reference in the above to evaluation of potentially useful aspects of a methodology, rather than a methodology as a whole, is not accidental. In fact mixing and matching particular features of different approaches to the needs of a particular type of application may be the best approach with the use of “traditional methods,” which were not per se created to address the issue of digital system modeling and therefore cannot individually be expected to cover in an acceptable way all the many facets of the issue. E.g., a traditional fault tree analysis may be adequate for the modeling of a relatively simple digital RPS logic, but a Markov model approach may be needed to address the fault handling features of a digital system software and redundant CPU architecture.
  - No matter what criteria are used, they should include the evaluation of methods in terms of whether they are effective at identifying and uncovering types of critical failure modes that have actually been observed in the operational experience of safety-critical digital systems.
    - In this respect, a categorization of types of systems in use and failures that have occurred should be adopted and/or developed and the suitability of methods to address the various categories should then be assessed.

Reviewer A also provided written comments after the meeting (see Attachment E).

### **Reviewer B**

- The problem statement significantly lacks clarity (i.e., stable regulatory environment vs. trying to become risk-informed vs. ....). For example, are the criteria for the regulatory

review of modeling? Hence, the conclusions of the report are not necessarily tied to the report's text or the regulatory premise (i.e., they are disjointed ideas).

- Doesn't necessarily disagree with conclusions, but the text does not support them. In particular, conclusions about the methods are not supported.
- The report starts by discussing regulatory items, and then moves into technical discussions.
- Each application reviewed using the report's criteria had different objectives. Hence, conclusions cannot be reached by comparing the extent to which the applications met the criteria.
- There is a confusion between what is meant by traditional and dynamic methods that needs to be clarified.
- Other methods for reliability modeling of digital systems may exist in other industries, and modified versions of them might be used for systems in the nuclear industry.
- May want/need to pursue a "blended" approach with the best features of traditional and dynamic methods (or more "advanced" traditional methods).
- Sophisticated methods may not be necessary because there is no good data, anyway.

### **Reviewer C**

- A link or relationship should be established between the criteria and other standards or procedures, such as the American Society of Mechanical Engineers (ASME) Level 1 PRA standard, and various guidelines on common cause failure (CCF).
- It is easy to implement modifications to digital systems, which can complicate modeling and quantifying data.
- Advanced reactors have many digital systems that perform control functions, as opposed to the reactor protection system (RPS) and emergency safety features actuation system (ESFAS) that are actuation systems. The former group needs detailed modeling, including considering many additional failure modes.
- Some modeling approaches reviewed may meet additional criteria, but it just wasn't documented.
- Some methods may only meet some of the criteria, but could play a role as part of the solution.
- The evaluation against criteria is limited by available information.

- A (quantitative?) method should be used for assessing software reliability. However, the resulting model does not have to be integrated with the overall PRA.

Reviewer C also sent written comments before the meeting (see Attachment F).

### **Reviewer D**

- Hardware (HW) and software (SW) reliability cannot be evaluated separately, otherwise HW/SW interactions cannot be captured (philosophical issue).
- Traditional methods and the proposed draft criteria don't necessarily capture all of the Type I and Type II interactions.
  - Type I - dependencies due to communication through the controlled/monitored process
  - Type II - dependencies due to direct communication (e.g., networking, multiplexing, or hardware linkages)
- Fault tree/event tree (FT/ET) and Markov as "traditional" methods probably can't address all the issues of modeling digital systems
  - There probably is a need to add a "twist."
- Detailed models are needed. There is probably a need for different modeling methods for different applications. A graded approach should be used depending on system function (i.e., safety or control).
- Statistical dependencies between failure events may require Markov treatment.<sup>2</sup> FT cannot account for these dependencies. J. Dugan (University of Virginia) proposed a method that
  - Uses timed "AND" gates to model conditional occurrence of events given certain events have occurred
  - Doesn't cover process interactions
- Dynamic methods may be needed to model communication, as well as dynamic methods for certain portions of the PRA, and then map them back into the PRA.
  - However, dynamic methods may not be necessary to model systems such as the RPS and ESFAS.

---

<sup>2</sup> After the meeting, Reviewer D sent the following statement: "...an explanation within the context of the relevant statement would be when the sequencing of events lead to different consequences. Then the consequences would be statistically dependent on the precursor events. For example, if an event B in the precursor sequence cannot occur before the previous event A occurs, then  $P(B)=P(B|A)P(A)$ . The standard ET/FT approach will not account for the conditional in  $P(B|A)$  in the quantification process. However, Markov approach is not the only way such a dependency can be accounted for..."

- [Reviewer A]: One can have a general framework using traditional criteria, and identify areas that require other approaches. Considering the issues related to Type I and Type II interactions is another way of looking at the different types and levels of digital system implementations that can be found in real life applications.
- [Reviewer B]: The complexity/accuracy of modeling may need to be driven by where data is available.
- The scarcity of probabilistic data is a big concern.
  - [Reviewer A]: Developing methods/models will point us to what data is needed,
  - [Reviewer A]: ...and also how to test the system, since testing is the only good source of data (as opposed to generic databases).
- Uncertainties should be propagated through the model.
- Some statements and terms need to be more specific, e.g., what is meant by Markov modeling.

Reviewer D also provided written comments before the meeting (see Attachment G).

### **Reviewer E**

- Cybersecurity also might be an issue that should be addressed in the criteria.
- Timing can be included in ETs/FTs through a phased-mission analysis.
- Dynamic FT gates can be used to capture dependency and timing.
  - Reviewer E showed a book that used Dugan's method. [M. Sonza Recorda, Z. Peng, and M. Violante, Editors, "System-Level Test and Validation of Hardware/Software Systems, Springer Series in Advanced Microelectronics, 2005]
- MIL-HDBK-217 and PRISM have been superseded by 217 Plus.
- Even though an approach for software reliability analysis is included in NASA's PRA procedures guide, NASA has not agreed yet on an approach for analyzing software reliability.
- NASA's approach (dynamic flowgraph methodology [DFM]) may not qualify as a traditional method.
  - Reviewer A disagrees for several reasons: a) the NASA approach is not based on DFM, but on traditional event-tree / fault-tree modeling where possible, combined with traditional SW reliability estimation methods; b) the application example in the

NASA PRA Procedures Guide shows a traditional ET/FT analysis of a DMSP satellite attitude control system in combination with the Schneidewind SW reliability estimation method; DFM is used as an example of what can be done when more detailed dynamic modeling is necessary; c) DFM itself is documented in at least four NUREGs and two NASA reports, dating back to the mid-nineties.

- Level of detail:
  - Needs to be tied to purpose of analysis.
    - This is discussed in the report but not listed in the final criteria.
    - The level of detail need not capture design features that could affect unreliability if sufficient data are available to bound unreliability and that is the output needed for decision-making.
  - If system unreliability is dominated by components, such as circuit breakers and valves, there is no need to go to microprocessor level to ascertain it. .
    - System may also include operators who can over-ride failed digital controllers.
      - Controller failure ANDed with operator failure to over-ride.
    - This can limit level of detail needed, for example, need for controller FMEA.
- Software failures:
  - Cannot understand Criterion 3.2 for software (the reference [Chu, 2006b] is not available for review).
  - May not need logic models for software; depends on purpose of analysis.
    - It may be sufficient to bound software contribution.
- Modeling of dependencies:
  - Failure of communication network is important consideration.
    - Recent “data storm” at Browns Ferry is an example.
- Human errors:
  - Human reliability analysis also must consider operator recovery from failure of hardware/software.
- Probabilistic data:
  - Statement that digital hardware data are “scarce or non-existent” is probably too strong because some data are available.
  - Recent discussion with Honeywell (Netherlands) suggests there is a large amount of (non-nuclear) data on programmable logic controllers (PLCs).

- HW and SW data, but the latter may not be applicable.
- May need to analyze data for a particular application.
- It is not clear whether these data are publically available.
- Hardware data requirements should specifically address the Bayesian approach.
  - Use of “generic” data or allied-industry data as prior distribution.
  - Adjustment of data from other applications or environments.
  - Dealing with uncertain data (e.g., uncertainty in failure count).
- For software failure, testing data could be appropriately used in one of the software reliability growth models discussed in ANSI/AIAA Std. R-013-1992 and implemented in CASRE software.

### **Reviewer F**

- Challenges associated with modeling digital systems:
  - software reliability
  - common cause failures (including software)
  - hardware/software interactions
  - failure data
  - interfacing digital system models into a PRA
  - time dependencies
  - diagnostics/fault tolerance/coverage
  - failure modes (including unknown or unforeseen failure modes).
- Challenges associated with developing a review process consistent with current regulations/guidance:
  - level of modeling detail
  - acceptance guidelines
  - PRA quality -attributes for digital system modeling
  - open issues - use of PRA with a deterministic defense-in-depth philosophy/methodology.
- The objective of the criteria is to support regulatory decision-making, e.g., a decision on giving credit to certain fault-tolerant features.
- It is helpful to link the criteria to standards, such as the ASME PRA standard.
- In establishing evaluation criteria, it is not as easy as saying one needs to be able to adequately model the unique aspects of digital systems:
  - Evaluation criteria must reflect both the characteristics of digital systems and how they are used in nuclear plants.
  - A method must be developed for categorizing digital systems.

- The community needs to continually be looking at operational experience, for example:
  - EDG load sequencer failure at Turkey Point, and
  - Data storm at Browns Ferry.
- Operational experience will affect how the evaluation criteria should be written. For example, the Category 1 criteria on level of detail currently include the phrases “design features that affect reliability” and “at the microprocessor level.” System state and cross system inter-connectivity as failure modes have been observed, and whether they should be explicitly part of the criteria should be considered.
- A method of categorizing digital systems may need to be developed to help determine the level of modeling detail needed.
  - Consider the need for additional Category 2 criteria for determining the level of detail to be used in identifying failure modes. .
- In modeling software reliability, the concepts are often stated very differently than in traditional PRA terminology.
  - We should use terms that software people understand, or at least note the analogous terms (e.g., “operational profile” instead of “context” and “software-centric”).
- The criteria need to be more consistent and/or may need to be applied in a particular order, for example failure modes (Category 2), before CCF modeling (Category 4), before level of detail (Category 1).
- In the evaluation criteria for human errors (5.1 and 5.2)
  - Criterion 5.1 needs to be reworked to include the way humans introduce faults into the software.
  - Man-machine interface (MMI), or more appropriately human-system interface (HSI), is outside the scope of the system model.
- Hybrid analysis methods should be considered for developing applications. For example, NASA’s study of the International Space Station (ISS) used traditional FT/ET modeling for the most part, but Markov modeling for many digital systems.
- A possible definition of “traditional” method is one that is commonly used, well established, including large-scale applications.

### **A.3. TRADITIONAL METHODS AND THEIR APPLICATIONS**

The discussions focused on three main issues:

- What is the definition of “traditional” methods, and therefore, what methods should be included in identifying and selecting “traditional” methods?
- Are the conclusions stated in the report clearly supported by applying the criteria presented in the report?
- Should the proposed criteria be applied to the methods, or to the models/applications?

The panel members’ discussions on these three issues, as well as about alternative methods and applications, are summarized in the sections below.

#### **A.3.1 Defining and Identifying “Traditional” Methods**

Points of general agreement among the panel members: “Traditional” methods are difficult to define precisely. Separating methods based on “dynamic” versus “traditional” does not really help. Some methods can be considered traditional if they are used for a part of a model, but non-traditional if they are used for the whole model. Ultimately, binning traditional methods versus non-traditional methods includes subjective elements. Traditional methods will involve methods commonly used by the nuclear industry, since the NRC is interested in evaluating licensees’ submittals. Therefore, traditional methods can be defined as

- Methods that can be used in near-term (or somewhat near-term) to address at least some aspects of digital system reliability modeling and quantification.
- Methods that have had real-world application in the nuclear industry.
- Methods applicable to the kinds of decisions that the NRC will face.

Individual panel members had the following observations:

#### **Reviewer F**

The interactions and dependencies of digital systems may require methods that can overcome some of the limitations of traditional FT methods. Binary Decision Diagrams (BDDs) can help with coherence problems. Already, some applications of BDDs are used by the telecommunications and aerospace industries. In a Norwegian study, Dahll applied the method. Bayesian Belief Networks (BBNs) also may be helpful. Ali Mosleh developed a technique combining the BBN approach with BDDs. However, so far there are no large-scale applications of these methods to digital systems.

#### **Reviewer E**

Simulation methods, such as discrete event simulation (DES), also could be considered as traditional methods. In a 2003 Finnish paper, BBN was used to analyze the software of a relay. One could also question whether DFM qualifies as traditional. While the NRC has traditionally relied on FT/ET



models for reactor safety analysis, a contractor for the Office of Nuclear Material Safety and Safeguards (NMSS) used discrete event simulation in a medical application.

### **Reviewer D**

Traditional Markov techniques are not really that useful for evaluating digital systems. Depending on the definition of traditional methods, DES and BBNs could be considered as such. It is not clear that DES is practical. For simulation methods, the problems are that the sequence of failure modes cannot be captured, and integrating the results with a PRA is problematic. Dugan tries to capture data dependencies, employing a method she calls “Dynamic Fault Tree.”

### **Reviewer C**

The difference between traditional and dynamic methods is not clear, and it is hard to differentiate between them. For example, some methods would qualify as “non-traditional” if used to model an entire digital system, but would as “traditional” if used for specific aspects of the modeling, such as BBN or testing or simulation with fault injection. The only method everyone agrees is clearly traditional is the FT/ET approach.

### **Reviewer B**

EPRI supplies the R&R workstation that could be considered the most widely used software for reliability analysis. Methods included in the R&R workstation are currently traditional. However, new capabilities for the R&R workstation, such as BDDs and Declarative Modeling, will be released by the end of the year, and may include the ability to use phased-mission times. However, EPRI may not release this capability due to the concern that it may not be used properly, thereby distorting some plant’s risk profiles.

### **Reviewer A**

It is difficult to distinguish between traditional and dynamic methods. For example, BBN and Petri net methods can be considered traditional. Another example is using multi-value logic methods (predecessors of DFM), which have been employed in the chemical industry, but not for digital systems. It should be noted that reliability prediction methods are not really methods but a source of data. There is a NASA report<sup>3</sup> on an application of the software reliability quantification method described in the NASA PRA procedures guide. He elaborated on this subject in his written comments (Attachment E).

---

<sup>3</sup> This report is: “Risk-informed Safety Assurance and Probabilistic Risk Assessment of Mission-critical Software-intensive Systems,” AR 07-01, ASCA, June 2007.

### **A.3.2 Conclusions Stated versus Applications of the Criteria Presented in the Report**

Points of general agreement among the panel members: The report draws, or implies, conclusions about the models reviewed without knowing what original objectives the models were intended to satisfy. The report needs to further clarify that the models may not meet many of the criteria because they were not developed with the intent of doing so, but rather with objectives that may be quite different. Some models could have met more criteria if they had different objectives. The conclusions in the report should be more focused on the capabilities of the methods, not the capability of the models with respect to the criteria.

Individual panel members had the following observations:

#### **Reviewer B**

Some of the so-called “conclusions” in the report should be moved to the front to make the flow of the report more logical. The report reflects the order in which the work was done, but rearrangement could help with clarity, and produce a better report.

#### **Reviewer A**

The report should be portrayed as a demonstration of the criteria, not as judging specific methods or models. He elaborated on this subject in his written comments (Attachment E).

### **A.3.3 Criteria Applied to Methods or to Models/Applications**

Points of general agreement among the panel members: It would be desirable to have criteria to evaluate methods. However, since a digital-system model could involve combining methods to address different aspects of the model, one ultimately needs to apply criteria to the modeling. The report’s conclusions should be more focused on the methods’ capabilities, not that of the models. Criteria could be applied against applications and this information used to evaluate methods. In any case, to proceed with discussing individual criteria, it was generally agreed that criteria should be viewed as being model/application-oriented.

Individual panel members had the following observations:

#### **Reviewer B**

The project is proposing which methods to pursue. It will be seen how well they meet the criteria, based on the evidence from the two test cases to which they are applied. The next step is extrapolation, i.e., to extend the conclusions from the test cases to reach general ones that can be considered generally applicable to analyzing digital systems. It will be necessary to support this extrapolation unless it is a straightforward inference, i.e., unless it is obvious that a particular method could meet a particular criterion if applied for that purpose – this requires a judgment about whether substantial additional work would be necessary for the method to meet the criterion.

### **Reviewer C**

Some of the criteria presented are overly specific, such as requiring modeling the loss of HVAC ; they should be more general, not design-specific.

### **Reviewer E**

Ultimately, one needs to apply criteria to “modeling,” because the digital system model could involve a combination of methods to address its different aspects. .

### **Reviewer D**

If the capabilities/limitations of the methods are extrapolated based on the models, the work may be criticized as being speculative.

### **Reviewers A, B, and F**

The criteria can be used to evaluate the methods, and the models then used as evidence of the methods’ capabilities. Reviewer A elaborated on this subject in his written comments (Attachment E).

## **A.4. COMMENTS ON REVIEW CRITERIA**

### **A.4.1 General Comments on Review Criteria**

The following general comments were made about the review criteria:

- In general, the panel members felt that the criteria did a good job in covering the desired characteristics of digital-system reliability modeling.
- The state-of-the-art of reliability modeling cannot support all of the identified criteria, and additional research is needed, e.g., in reliability data, CCFs, and software reliability.
- It was recognized that the criteria have different levels of detail, degrees of specificity, and importance, and some criteria include not only review criteria but also background information. It is recommended that the criteria are made more succinct, and that supporting rationale, examples, and guidance on how to satisfy them is moved to the background discussion.
- Some criteria are similar to those in PRA standards, e.g., the ASME Level 1 PRA standard. Any relationship to the ASME standard should be stated.
- The criteria should be general without specifying the methods that should be used. For example, failure modes and effects analysis (FMEA) is only one of the methods used for identifying failure modes (others include hazard analysis, and hazard and operability study [HAZOPS]).
- A consistent terminology should be developed for all of NRC's projects dealing with digital systems.

The rest of this chapter summarizes the discussions on the categories of review criteria, in the same order they occurred during the panel meeting (Attachment H gives the order in which the review criteria categories/subcategories were discussed). For each criteria category, there is a brief description of the category, followed by any general comments agreed-upon by the members of the expert panel, and the comments from individual experts.

### **A.4.2 Category 1: Level of Detail of the Model**

While the criteria in the other categories represent a collective set of criteria for evaluating digital-system reliability models, the three criteria in this category are mutually exclusive ones at different levels of detail (i.e., a model would only be expected to meet one of the three criteria). Criterion 1.1 represents the ideal level of detail of a model, Criterion 1.2 represents the level of detail that the authors believe is reasonably achievable, and Criterion 1.3 represents the level of detail of the digital-system models included in the design certification PRAs for new reactors (i.e., AP1000 and the Economic Simplified Boiling Water Reactor).

The panel thinks that the level of detail of modeling should depend on the study's objective, , and recommended that the existing proposed criteria in this category be replaced by the following alternative criterion: "Modeling should reflect all significant failure modes (functional and

physical), be developed to the level of detail of supporting information, and provide output needed for risk-informed decision-making."

The comments of individual panel members are provided below:

- [Reviewer A]: Criterion 1.2 makes an assumption and is problematic. It should instead indicate that the model should address both the physical and functional characteristics of a digital system, e.g., the timing of a central processing unit (CPU) failure can affect the type of impact the failure may have at the plant level.
- [Reviewer B]: Criterion 1.1 is the only criterion; Criterion 1.3 is an exception, and Criterion 1.2 is a very specific criterion that can conflict with Criterion 1.1. In Criterion 1.3, "...can adequately support the objective of the modeling" should be replaced with "...capture all dependencies including software."
- [Reviewer C]: The final criterion should address "operational and functional characteristics". Since a circuit board may perform several functions, physical and functional features cannot be separated.
- [Unknown]: It is necessary to consider the difference between "functional" and "physical" failures.
- [Reviewer E]: An alternative criterion should be used, based on the panel discussion: "Modeling should reflect all significant failure modes (functional and physical), be developed to the level of detail of supporting information, and provide output needed for risk-informed decision-making."

#### **A.4.3 Category 2: Identification of Failure Modes of the Components of Digital Systems**

BNL indicated that from their experience it is very difficult to undertake an FMEA of digital systems and little guidance is available. For example, what is the level of detail at which an FMEA should/can be performed (subject to limitations on design detail and knowledge)? Are the failure modes realistic and complete? For example, can an output bit being stuck high be an isolated failure mode, knowing that the bit is physically connected to other parts of the system?

The panel thinks that a criterion should not advocate a particular method, i.e., FMEA, and that other methods also can be used, e.g., HAZOPS. The title of the category should be changed to include "components of" before "digital system." An alternative to Criterion 2.1 was proposed: "A technique for identifying failure modes of the basic components of a digital system, and their impact on the system, should be applied." The discussion about Criterion 2.3 generated a recommended new criterion that the "...failure modes that have occurred in operating experience should be examined." For example, important software failures have occurred as a result of problems with requirement specifications. The panel also recommended rewording Criterion 2.3 and including it as a sub-bullet to the new criterion.

The comments of individual panel members follow :

- [Reviewer D]: By design, FMEA is intended to identify immediate impacts, not to model fault propagation. In the nuclear field, FMEA is a precursor to fault trees, i.e., it considers

the immediate impact of component failure, not the systemic impact. Defining failure mode according to functions might be wrong. Criterion 2.1 may be neither feasible nor necessary. Arbitrary output as a failure mode should be considered.

- [Reviewer A]: An FMEA (more so when applied in FMECA – Failure Modes, Effects and Criticality Analysis) normally considers the effects and consequences of a postulated failure. Military Standard 1629 and Handbook 338b provide guidance on FMEA.
- [Reviewer C]: FMEA should be related to deterministic criteria; other methods can be used to identify failure modes, e.g., hazard analysis and HAZOPS.
- [Reviewer F]: Standard Review Plan (SRP) Chapter 7 addresses Criterion 2.3. Failure modes should not be screened at this stage because their effects in combination with other failures have not been identified. The model should allow the possibility of design errors.
- [Reviewer E]: FMEA usually is done by “designers,” and is a good starting point for system analysis. Fault trees can be used to model multiple failures.
- [Reviewer B]: The industry performs FMEA routinely, and General Public Utilities has guidance on FMEA that is not specific for digital systems. Criterion 2.3 should not be here.
- [Reviewer F]: Operational experience suggests that digital systems are vulnerable to faults associated with diagnostic features.

#### **A.4.4 Category 3: Software Failures**

Criterion 3.1 suggests that the contribution of software failures can be considered using either a “software-centric” or “system-centric” approach, while Criteria 3.2 to 3.4 apply only to the “software-centric” approach. Criterion 3.2 associates the occurrence of software failures with that of triggering events, and requires a model of software failures that is consistent with this concept. Criterion 3.3 suggests separately considering the application software and support software, including the operating system and platform software. Criterion 3.4 emphasizes the importance of exploring the context wherein a piece of software is challenged, and states that a quantitative software reliability model should be able to account for different contexts.

The panel discussed Criterion 3.2 extensively, and agreed that the term “triggering event” must be explained in the background discussion. The panel also agreed that separating application and support software, as indicated in Criterion 3.3, is important because of their differing amounts of operating experience. In addition, the panel recommended listing Criteria 3.2 and 3.4 next to each other.

The comments of individual panel members are given below:

- [Reviewer B]: Many criteria are overly wordy, i.e., they should simply state the criterion, and the justification or “how to” should be moved to the background discussion.
- [Reviewer F]: The terminology of “system-centric” and “software-centric” can lead to misunderstanding.
- [Unknown]: Criterion 3.2 is too specific on “how to.” There are other approaches for quantifying software reliability, such as parametric and non-parametric methods. The way Criterion 3.2 is stated suggests there are hidden assumptions pointing at a certain direction,

and should be removed. Instead, it should state that the model needs to articulate how it arrives at its failure rates or probabilities.

- [Reviewer A]: Regarding Criterion 3.2, the software of a control system may have a conditional failure probability that if combined with a rate of occurrence of some condition becomes a failure rate.
- [Reviewers D and F]: Criterion 3.2 could be worded “...don’t consider software failures in the abstract, consider them in the context of the system.”
- [Reviewer F]: Regarding Criterion 3.2, there are other ways to get software failure besides just “triggering events” (e.g., bit drops, specification errors, hardware/software interface errors).
- [Reviewer F]: Criterion 3.3 should include the interactions between applied software and support software, and between software and hardware. Due to the potential for CCF, there may be a need to consider some software development tools that generate application software. Commercial off-the-shelf (COTS) and communication software should be considered.
- [Reviewer C]: Backbone (support) software and software development tools should be considered.
- [Reviewers A and F]: Criterion 3.4 is just an extension of Criterion 3.2.
- [Reviewer B]: Criterion 3.2 is on triggering events, and Criterion 3.4 is on functions; both are needed.

#### **A.4.5 Category 4: Modeling of Dependencies**

The criteria in this category consider different types of dependencies that should be accounted for in developing a reliability model of digital systems.

The panel agreed that Criterion 4.2 (Common Cause Failures) is a “catch-all” bin for dependencies that are not explicitly modeled; therefore, it should be moved to the end of this category and will be discussed last.

The comments of individual panel members are shown below:

- [Reviewer B]: The criteria essentially are a list of dependencies. One doesn’t need to specify separate criteria for each dependency (that wasn’t done elsewhere, e.g., specifying devices). Reviewer B recommends including the dependencies as a bulleted list instead of as separate criteria.
- [Reviewer F]: The subcategories are “uneven.”
- [Reviewer D]: Type I dependencies (interactions with physical processes) are not clearly captured. They should be added, maybe under Subcategory 4.5. The time constant of the system changes, and the response times differ. The coupling (with physical processes) may become much tighter. Uncertainty in discrete sampling times may lead to system failure, even if the Nyquist criteria are satisfied; this is unique to digital systems.
- [Reviewer C]: Self-testing differs from other fault-tolerant features, and should be modeled separately (though not necessarily in this category).

- [Reviewers B and E]: The dependency breakdown may be too fine, and may result in the summation of a large set of overly conservative (uncertain) values. It may be preferable to lump more of the individual dependencies into the “catch-all” CCF bin, based on current state-of-knowledge.
- [Reviewers A, B, and D]: It is only possible to model to the level of the state-of-knowledge. However, this is an evolving boundary because there even is uncertainty about what is the current state-of-knowledge.
- [Reviewer F]: It is important to explicitly model aspects of the system that risk-informed applications are trying to address; therefore, many of these dependencies may need to be explicitly modeled.

#### **A.4.5.1 Subcategory 4.1: Communication Networks/Buses**

Components of digital systems are interconnected through buses, hardwired connections, and communication networks. Through the connections, information is exchanged and used in calculations and decision-making. The criteria in this sub-category address modeling of failures associated with the connections of components of digital systems at different levels (e.g., intra-channel communications and inter-channel communications).

The panel agreed that the Browns Ferry data storm incident is a good example indicating the importance of modeling communication-related failures.

The comments of individual panel members are provided below:

- [Unknown]: The lengthy explanation should be eliminated from Criterion 4.1.1 to make it more succinct.
- [Reviewer B]: The criteria should be kept broad so people do not leave things out.
- [Unknown]: A distinction should be made between the four criteria in this sub-category.
- [Reviewer E]: The meaning of communication network is unclear.
- [Reviewer B]: Any available operating experience should be included in the discussion, but it must be emphasized that examples and operating experience are not the “end all and be all.”
- [Reviewer E]: It is not clear what is really meant by “failure of the communication network.”

#### **A.4.5.2 Subcategory 4.3: Support Systems**

The criteria in this subcategory are applicable to support systems that are shared by the components of the digital system and the rest of the plant, as modeled in a PRA. They ensure that the dependencies are properly accounted for when the model is integrated with the PRA.

There are no general comments on the overall category. The comments of individual panel members appear below:

- [Reviewer F]: Power quality and power availability issues also should be considered in the criteria of this subcategory. Other issues in 50.49 or RG 1.97 and the issue of



radio-frequency interference should also be included. A distinction should be made between the support systems that should be modeled and the data that capture the impacts of support systems.

- [Reviewer B]: These criteria are an example of being too specific. They imply that the only support- system dependencies that need to be addressed are the two specific dependencies identified in them. It is better to have general criteria and provide examples in the discussion, making it clear that these are only representative examples.
- [Reviewer D]: The HVAC in Criterion 4.3.2 should be generalized as "operating environment."
- [Reviewer A]: Developing overly specific criteria makes it very unlikely that any application can address them all.

#### **A.4.5.3 Subcategory 4.4: Sharing Hardware**

This criterion is intended to emphasize that some digital systems may be implemented by running different software on the same digital hardware, and this dependency has to be modeled.

There are no general comments on the overall category. The comments of individual panel members follow :

- [Reviewer F]: This subcategory should focus on shared digital components, i.e., chips. There should be more discussion in the rationale on what is meant by shared hardware (e.g., sensors, multiplexers, and voters) that should indicate what types of dependencies are addressed, and which ones must be modeled explicitly. It is also important to consider specific possible regulatory decisions about putting RPS and ESFAS on the same hardware platform, or assuming what systems fail in the diversity and defense-in-depth analysis.
- [Reviewers A, D, and F]: The statement on RPS/ESFAS modeling in Criterion 4.4 should be removed. The phrase, "e.g., by linking fault trees" is too specific, and should also be removed from this Criterion.
- [Reviewer D]: Another type of sharing hardware could be the data sharing by processes of software. Two processes might try to access data in a storage device simultaneously, causing data race and system lock-up.

#### **A.4.5.4 Subcategory 4.5: Interactions of Digital Systems with Other Systems**

The intent of the criteria is to ensure that the interfaces between the digital system and the rest of the plant are properly accounted for.

There are no general comments on the overall category. The comments of individual panel members are provided below:

- [Reviewer D]: Type I interactions, i.e., devices communicating through the controlled/monitored process, are not captured by the criteria in this subcategory. An additional criterion that reflects this interaction might be needed.

- [Reviewer F]: Loosely coupled dependencies are also not captured by this subcategory criteria; an example is a low reactor pressure signal for reactor trip. Definitions of loosely coupled dependencies and tightly coupled dependencies are given in C. Perrow's book [Normal Accidents, Living with High-Risk Technologies, Princeton University Press, Princeton, New Jersey (1999)] or Steven Arndt's paper ["Development of Regulatory Guidance for Risk-Informing Digital System Reviews," NPIC&HMIT 2006].
- [Reviewer B]: The meaning of "incorrect sensor input" in Criterion 4.5.1 is not clear. Generally, the sensor inputs go to digital systems only, not to other components or systems.
- [Unknown]: The meanings of systems and channels are not clear. It may be better to use systems and sub-systems.

The following comments were made on Criterion 4.5.2 that addresses the modeling of voters and other logic devices:

- [Reviewer F]: It is too specific and should be either dropped or modified.
- [Reviewer D]: It should be modified to be made more general by removing the reference to specific devices.
- [Reviewer C]: It has already been covered and can be dropped.
- [Reviewer B]: It should be dropped.
- [Reviewer A]: It should be modified.
- [Reviewer E]: It should either be dropped completely or included in the supporting rationale.

#### **A.4.5.5 Subcategory 4.6: Modeling of Fault Tolerance Features**

The criteria are based on known weaknesses of models of digital systems, and intended to ensure that they are avoided. Criterion 4.6.1 considers identifying the failure modes that can be detected. Criterion 4.6.2 concerns the potential of doubly crediting fault coverages. Criterion 4.6.3 considers modeling the dependency on fault tolerant features. Criterion 4.6.4 relates to modeling failures to properly cope with detected failures.

There are no general comments on the overall category. Individual panel members' comments are given below:

- [Reviewer F]: Fault tolerance includes design and tests for fault tolerance. A high-level criterion should state that fault tolerance design features should be modeled explicitly, including all ways in which they function, and both their potential positive and negative impacts. Some specific information can be included in sub-bullets and sub-criteria.
- [Reviewers C and F]: Fault identification features and continuous self-test features should be distinguished.

#### **A.4.5.6 Subcategory 4.2: Common Cause Failures**

The criteria specify that hardware and software CCF should be modeled within a channel if redundancy exists, between redundant channels, and between digital systems. It also was pointed out to the panel that some people from industry are claiming that "...digital hardware failures often

can automatically be detected and possibly alarmed; therefore, it is very unlikely that two hardware failures take place at exactly the same time.”

The panel recommended that "identical" in Criterion 4.2.2 should be changed to "common," or "similar," or "very similar." Also, there should be more discussion in this subcategory on its "residual" nature. An example statement is "The previous criteria in this category addressed dependencies that should be explicitly included in this model. This criterion is intended to address other potential dependencies that are not explicitly modeled."

The comments of individual panel members are provided below:

- [Reviewer A]: "Intra-system" and "inter-system" should be combined into the same criterion.
- [Reviewer E]: Consideration of inter-system CCF may be pushing the envelop beyond the state-of-the-art, since these types of failures are not currently included in other system models in PRAs.
- [Reviewer B]: It may not be credible for two different digital systems to be subjected to the same trigger mechanism.
- [Reviewer E]: If someone tries to model inter-system CCF for digital systems, there will be no data. They will generate a conservative model that might distort the plant’s actual risk profile, .
- [Reviewer B]: The inclusion of the inter-system CCF criteria could entail spending much money for analysis and possibly redesign, all for nothing.
- [Reviewer F]: Questionable criteria should be included for now, and can always be eliminated later.
- [Reviewer A]: The timing aspects of potential common cause failures may preclude the concern over inter-system CCF.
- [Reviewers A and B]: It makes sense to include exceptions or “wriggle-room” in the criteria, e.g., including the phrase "...or else demonstrate that it does not have to be included."
- [Reviewer F]: We may want to say "should be considered" instead of “should be modeled.”

#### **A.4.6 Category 7: Probabilistic Data**

Probabilistic data for both hardware and software failures are needed to quantify reliability models of digital systems. While component-specific data are preferred, generic data also can be used if there are no specific data, and the generic data are properly collected. The same consideration applies to CCF parameters and estimates of fault coverage.

There are no general comments on the overall category. Individual panel members’ comments follow:

- [Reviewer A]: Regarding Criterion 7.1, it is very unlikely that truly component-specific data will ever be available at the desired level of statistical confidence and fidelity, due to the short design cycles of digital hardware components (i.e., digital systems are upgraded very quickly).

- [Reviewer B]: Many of Criteria 7.1 - 7.10 (for hardware data) are good practices, but not necessarily qualified to be criteria. Thus, Criterion 7.8 about the CCF data is particularly problematic since generally plant-specific CCF data cannot be obtained.
- [Reviewer F]: Regarding the criteria in this category, referring to the good practices of the ASME PRA standard or the data handbook is suggested. However, unique practices specifically related to digital systems should be emphasized. An example is whether the data for the same card, or similar versions of cards, are used in the reliability model, and whether the difference in data sources matters.
- [Reviewer E]: Replacing "data" with "information" is suggested. It would be good to provide some guidance on how to use generic data (e.g., from Military Handbook 217), and on how to account for uncertainty.
- [Reviewer B]: EPRI can talk with new reactor vendors about starting an effort to build a digital instrumentation and control (I&C) database.
- [Reviewer D]: Criterion 7.11, "quantifying the software failure probabilities" suggests using a software-centric model, but this does not necessarily have to be the case.
- [Reviewer F]: Some of the intrinsic issues in Criterion 7.11 associated with software reliability data, should be highlighted, such as software revisions, software evolution (e.g., conglomerating operational data, test data), and other issues well-known in the software-reliability community.
- [Reviewer D]: Criterion 7.11 can be reworded to "A method for quantifying the contribution of software to digital system reliability should be used and documented."
- [Reviewer F]: Since Criterion 7.11 is data-related, the word "contribution" might not be enough, and this criterion may need re-wording.
- [Reviewer E]: Having to come up with a value for failure probability is a concern. An alternative choice could be "bounding" the failure probability instead of "quantifying."
- [Reviewer A]: An encouragement to develop quantitative estimates of the failure probability of digital systems is appropriate, otherwise everyone will stick with the current status quo, by which arbitrary assumptions of perfect digital system reliability are often made without justification.

#### **A.4.7 Criteria 8.4 and 8.5: Uncertainty**

Uncertainty criteria include both model uncertainty and parameter uncertainty. With the state-of-the-art in modeling digital systems, it is important to consider uncertainty.

There are no general comments on the overall category. The comments of individual members are provided below:

- [Reviewer F]: The criteria could refer to some literature on uncertainty in software modeling, which can be provided after the meeting.
- [Reviewer B]: For Criterion 8.4, it is not clear how many and which alternative assumptions should be discussed and documented. In some EPRI guidance, only those alternative assumptions that impact the CDF by a factor of two or more are required to be documented. Another issue is how to conduct the uncertainty analysis.

- [Reviewer B]: For Criterion 8.5, the point estimate may be the only value used in many applications. Therefore, propagation of uncertainties is not necessarily possible in these instances.
- [Reviewer F]: There is an Office of New Reactors' (NRO's) project on using sensitivity studies for digital system screening criteria that may be available by the end of this summer.

#### **A.4.8 Category 6: Ease of Integration with a PRA Model**

Most PRA models are built using the fault tree/event tree method. Thus, it is desirable to build the reliability model of a digital system in such a way that it can be integrated into the existing PRA framework.

There are no general comments on the overall category. Below are the comments of individual panel members:

- [Reviewer D]: There is a procedure for converting Markov model results to cutsets that can be implemented in a software.
- [Reviewer F]: Criterion 6.1 should be modified. An alternative criterion for it is, "For the digital system reliability model to be fully effective, it should be possible to integrate it into the plant PRA model. The process for integrating the model should be relatively straightforward so that it can be mechanized through software and can be easily verified."
- [Reviewer D]: The title of this category is questionable. The ease of accomplishing something is subjective and differs from person to person. "Ease" should not be considered a criterion.
- [Reviewers A and E]: Both suggest removing "ease" from the title and criterion.
- [Reviewer F]: The word "ease" can be removed from the title and criteria, but the concept of "ease" should be included in the discussion or rationale. The discussion should also address the issue of the process being "mechanized," as mentioned in his previous comment above.
- [Unknown]: In the discussion, refer back to all of the modeling features that were previously described..
- [Reviewer C]: The digital system reliability model should be compatible with the PRA model, i.e., should avoid other means of arriving at a likelihood of digital system failure that may not be compatible with the established PRA framework.
- [Reviewer C]: Another aspect of integration is that the failure of a digital system may generate an initiating event with possible additional failures of mitigation features. This should also be integrated with the PRA model.

#### **A.4.9 Category 5: Human Errors**

Generally, human errors related to digital systems can be treated in the same way as analog systems. They mainly are due to two factors: errors introduced during upgrading digital systems, and errors related to the MMI (man-machine interface).

There are no general comments on the overall category. The comments of individual panel members are set out next:

- [Reviewer F]: Criterion 5.1 is fine. It is probably worthwhile to add some caveats or sub-bullets to illustrate it. In Criterion 5.2, MMI should be replaced with HSI (human-system interface). However, HSI issues are beyond the scope of modeling digital system reliability, except for the dependency of human reliability analysis (HRA) on the state of the digital systems.
- [Reviewer D]: You can not model the human errors associated with a digital feedwater control system in a traditional PRA because there are so many interactions.
- [Reviewer C]: An HRA associated with digital I&C is not simple, especially recovery actions. Manual actions depend on the design. From an MMI perspective, consideration should be given to how the operator will recognize the situation with a particular set of process signals.

#### **A.4.10 Criteria 8.1 - 8.3: Documentation and Results**

These criteria consider documentation of key assumptions and results.

There are no general comments on the overall category. The comments of individual panel members are provided below:

- [Reviewer B]: In Criterion 8.1, a qualifier should be applied to the word "assumptions" as a condition for needing to be documented, e.g., "key" or "unique" assumptions related to digital systems. Authors may refer to the ASME PRA standard or RG 1.200 errata, which will be available soon, on uncertainties and assumptions.
- [Reviewer F]: In Criteria 8.1 and 8.2, the term "logic model" is too specific and should be replaced with "model."

## **A.5. CONCLUDING REMARKS**

Two members said that they discussed everything they wanted to at the meeting, and had nothing more to add. The concluding remarks of other members are summarized below:

### **Reviewer B**

The report is a good product and contains valuable information. Its purpose should be further clarified. It is necessary to make sure that the terminology and criteria is consistent through the whole report. Another issue for clarification is whether the criteria were developed to evaluate the modeling or the methods, as this will affect the conclusions and recommendations on method selection presented in the report, which are not supported by the text. The report should better capture the authors' philosophy and intent, since such knowledge might lead readers directly to the criteria. Sometimes, this is not always given in the report and the criteria are not easy to follow. It is also recommended that criteria be consistent with the Level 1 PRA standard. Generally, data issue and the treatment of software are big challenges, and are not expected to be resolved in this document; however, they should be considered. A good starting point on the issue of data is to collect it from other industries or sources named by panel members. Currently, the failure modes of digital systems are based on FMEA or operational experience. It is desirable to have a means to systematically identify the failure modes of digital systems.

### **Reviewer C**

Although the report mentioned that the criteria were developed for specific applications, Chapter 4 in this report is trying to evaluate methods. It is not clear how to fix this, but it definitely needs some clarification. Also, the report clearly suggests that some criteria are mutually exclusive. However, this is not discussed in rating the individual methods in Chapter 4. It is expected to have certain impacts on the rating. Rating scores should be downplayed because this information is misleading. Readers might think something is wrong with digital systems that already have been used in the nuclear industry since so many criteria cannot be met, as indicated in the report.

### **Reviewer F**

Other potential modeling methods should be articulated somewhere in the report, though it is not necessary to carry them throughout the report. The inclusion is suggested of a discussion of why some methods, such as CES, BBN, DFM, and various hybrid methods were selected for assessment in this report. The report needs restructuring so that model evaluation can be considered as supporting evidence of method evaluation, as previously suggested. Terminology and assumptions need to be better standardized for the NRC programs. A convergence of the philosophy and terminology between the PRA-world and the software-world also is desired; this is a bigger issue than this project. Operational experience, knowledge of how systems work, and their characteristics should be highlighted as part of the strategy for developing criteria.

## **Reviewer A**

The purpose and product of the report should be restated in a different light. After discussion, it is clear that the real objective is to arrive at criteria that could evolve into regulatory review criteria and to try to match these to what is judged feasible. This has not been articulated clearly in the report. Some better words in writing will be provided. There exists the appearance of potential conflicts between some of the current criteria. While they might not be actual conflicts, in certain cases some criteria should include a supporting rationale to explain how they complement other criteria, e.g., that the known CCF mechanisms complement the unknown CCF. It is not clear whether this applies elsewhere in the report, but a systematic review might identify these cases (e.g., using a Venn diagram).



## Attachment A

### Agenda of External Review Panel Meeting on Selection of Traditional Methods for Reliability Modeling of Digital Systems Brookhaven National Laboratory May 23-24, 2007

#### Day 1

Start Time	Duration (minutes)	Topic	Speaker
8:30	5	Welcome	Lehner, BNL
8:35	45	Background, overview, and objectives	Kuritzky/Siu, NRC
9:20	70	Presentation of preliminary comments by each member of the panel (10 minutes per member)	Members
10:30	15	Break	
10:45	75	Discussion on identifying “traditional” methods and their relevant applications	Panel
12:00	60	Lunch break	
1:00	120	Discussion on each review criterion: 1. Modification/deletion/addition 2. Limitations of the state-of-the-art and recommendations for additional research	Panel
3:00	15	Break	
3:15	105	Discussion on each review criterion: 1. Modification/deletion/addition 2. Limitations of the state-of-the-art and recommendations for additional research	Panel
5:00		Adjourn for the day	

**Day 2**

<b>Start Time</b>	<b>Duration (minutes)</b>	<b>Topic</b>	<b>Speaker</b>
8:30	90	Discussion on each review criterion: 1. Modification/deletion/addition 2. Limitations of the state-of-the-art and recommendations for additional research	Panel
10:00	15	Break	
10:15	105	Discussion on each review criterion: 1. Modification/deletion/addition 2. Limitations of the state-of-the-art and recommendations for additional research	Panel
12:00	60	Lunch break	
1:00	60	Discussion on each review criterion: 1. Modification/deletion/addition 2. Limitations of the state-of-the-art and recommendations for additional research	Panel
2:00	45	Concluding remarks	Members
2:45	15	Next steps and action items	NRC/BNL
3:00		Adjourn	

## **Attachment B**

### **List of Documents Sent to Panel Members**

1. Brief summary of the USNRC Office of Research's digital risk research program.
2. External peer review meeting description and agenda.
3. Executive summary and Chapter 6 of the National Research Council report (The full report is available at [http://www.nap.edu/catalog.php?record\\_id=5432](http://www.nap.edu/catalog.php?record_id=5432)).
4. Paper by Arndt, Siu, and Thornsby on "What PRA Needs from a Digital System Analysis" from PSAM6.
5. Draft BNL letter report: T. L. Chu, G. Martinez-Guridi, M. Yue, and J. Lehner, "Probabilistic Modeling of Digital Systems at Nuclear Power Plant: Traditional Methods Selection," Brookhaven National Laboratory, Draft Letter Report, April 2007.

## **Attachment C**

### **Expert Panel Meeting Attendees**

Expert Panel Members (in alphabetical order):

Tunc Aldemir (Ohio State University)  
Steven Arndt (USNRC)  
Ken Canavan (Electric Power Research Institute)  
Sergio Guarro (ASCA)  
Dana Kelly (Idaho National Laboratory)  
Taeyong Sung (Canadian Nuclear Safety Commission)

Facilitator:

Nathan Siu (USNRC)

NRC Project Manager:

Alan Kuritzky

BNL Authors:

Tsong-Lun Chu  
Gerardo Martinez-Guridi  
Meng Yue  
John Lehner

## **Attachment D**

### **Biographies of Panel Members**

#### **Tunc Aldemir**

Tunc Aldemir received his PhD in nuclear engineering from the University of Illinois and is a Professor of Nuclear and Mechanical Engineering at The Ohio State University. His broad area of specialization is nuclear reactor safety. His research in reliability and probabilistic risk assessment (PRA) focuses on systems that may be difficult to model using conventional techniques. He has published more than 80 refereed articles on dynamic methodology development for the reliability modeling of such systems.

Currently, Dr. Aldemir is involved in developing methodologies that will allow quantifying the risk impacts of upgrades of the digital I&C system in nuclear power plants and in developing computational tools to automate Level 2 PRAs and perform seamless Level 1-2-3 PRAs. He is a Fellow of the American Nuclear Society and on the editorial board of Reliability Engineering and System Safety.

#### **Ken Canavan**

The bulk of Mr. Canavan's 20 plus years of experience is in application of risk technology with the utility sector of the nuclear power industry. Mr. Canavan began his nuclear career at Toledo Edison's Davis-Besse nuclear power station, where he was involved in all aspects of the development of the plant specific probabilistic Risk Assessment (PRA). At GPU Nuclear, Mr. Canavan was a lead risk analysis engineer working on the Three Mile Island and Oyster Creek nuclear generating station risk management programs.

Following consultant experience as Manager of Risk Analysis for Data, Systems and Solutions SAIC and a Supervisor at ERIN Engineering, Mr. Canavan joined the Electric Power Research Institute (EPRI). Mr. Canavan is currently the Program Manager of the Risk and Safety Management (RSM) and Nuclear Asset Management (NAM) programs.

Mr. Canavan's experience is primarily in the area of risk technology and safety analysis. Specialty areas include methodology and tools development and unique applications of risk technology. Over his 20+ year of service, Mr. Canavan has participated or led the peer reviews of approximately a dozen large scale applications of risk technology within the nuclear and aerospace industries.

## **Taeyong Sung**

Taeyong Sung is a PSA and reliability technical specialist in Canadian Nuclear Safety Commission (CNSC).

He began his PSA career from 1989 in Korea Atomic Energy Institute with B.S. and M.S. degrees in nuclear engineering from Kyung Hee Universities in Korea. Since then he performed various areas of Level 1 PSA including digital I&C system reliability analysis.

He performed digital I&C system analysis for CANDU reactors as well as PWRs and leaded research projects to develop a PSA methodology for digital I&C system in NPPs for years in Korea.

In 2002, he joined Atomic Energy Canada Limited in Canada and he has worked in Canadian Nuclear Safety Commission (CNSC) since 2003. He is reviewing various risk and reliability analyses and developing regulatory documents and involving a research project for digital I&C system quantitative analysis.

Biographies of the other panel members are not available.

## **Attachment E**

### **Written Comments Provided by Reviewer A**

#### **Review Comments on Brookhaven National Laboratory Draft Letter Report “PROBABILISTIC MODELING OF DIGITAL SYSTEMS AT NUCLEAR POWER PLANTS: TRADITIONAL METHODS SELECTION”**

##### **1. Introduction**

This document provides comments that address key issues concerning the subjects covered by the BNL (Brookhaven National Laboratory) cited in the title, as well as related issues that emerged at the expert panel meeting held at BNL on May 23 and 24, 2007.

The intent of the comments provided is to assist the authors of the BNL report, who are in the process of completing and improving its content before publishing it in its final version. The subjects discussed hereinafter are a selected subset of topics that have been addressed verbally at the above-mentioned expert panel. However, this subset is covered again here in a more in-depth and organized fashion, primarily because it includes topics that the reviewer believes to have special relevance and/or have not been fully addressed at the meeting. Specific recommendations have been formulated and are offered in this review, as a possible solution for the most pressing and critical issues associated with the reviewed subjects.

The following primary areas of the BNL draft letter report are addressed in the following:

- Report objectives and their reflection in the report contents
- Report review of NASA reliability models
- Report conclusions and recommendations

Comments on the criteria for evaluation of digital systems models are not included in this review, because they were the more specific target of the expert panel discussions held at BNL on May 23-24, 2007. As such, they are extensively and more than sufficiently addressed by all the comments provided by the panel experts, and these comments are well documented on the BNL written compilation of the comments.

The principal findings of this review and the key recommendations that these findings suggest are summarized upfront, along with pointers to the sections of the comment text that provide the supporting rationale. Each section also lists at the end the specific recommendations that pertain specifically to the subjects discussed in that section.

##### **2. Summary of Findings and Recommendations**

The main findings of this review and related recommendations are summarized in the following. The section number identified after each finding or recommendation refers to later sections of this review where the reader can find a more detailed explanation of the finding/recommendation itself.

#### Findings:

1. A key objective of the BNL study is the identification of criteria for evaluation of future digital systems risk models and assessments in support of regulatory decision processes (Section 3).
2. The distinction between “traditional” and “advanced / dynamic” methods is not always clear (Section 3).
3. The Report objectives that refer to the evaluation of “models” and “methods (2 and 3 in BNL report Section 1.1) lead to practical contradictions both within the report evaluations and conclusions and with respect to what emerged in regard during the expert panel discussions on May 23 and 24, 2007 (Section 3).
4. The BNL reviewers have not been able, for lack of information or other factors, to develop a good understanding of the NASA “conditional risk method,” both as a framework for digital systems and software integration into PRA, and in terms of the application examples contained in the NASA PRA Procedures Guide (Section 4).
5. Some of the conclusions of the BNL report concerning “methods” do not seem to be supported by the evidence gathered in the course of the evaluations conducted on the “models.” Other methods appear to have been excluded from consideration primarily because examples of application (i.e., “models”) were not easily accessible by BNL (Section 5).
6. The BNL authors appear to be more oriented towards the development of hard to produce and update generic databases of digital system failure data, than on the development of component-specific test-oriented assessment methodologies. (This is an indirect deduction by the reviewer, based in equal measure on what is stated and what is not stated in the report and its conclusions and recommendations) (Section 5).

#### Recommendations

1. A better distinction and definition of “traditional” and “advanced / dynamic” methods should be provided upfront in the report. One such definition is offered in Section 3.
2. The Report objectives that refer to the evaluation of “models” and “methods (2 and 3 in BNL report Section 1.1) should be reconsidered. More specifically it is recommended that:
  - a. Objective 2 be reformulated in terms of pursuing the trial application of evaluation



criteria to available models (Section 3).

- b. Objective 3 be reformulated in terms of seeking the identification of key useful features of existing methods that may be assembled within an overall “hybrid” and flexible framework (Section 3).
3. The BNL assessment of the NASA PRA Procedures Guide approach and framework should be revisited in light of the more recent and detailed information that is now available (Section 4). Particular attention should be given to this approach as an example of “hosting framework” for a hybrid combination of methods and models (see also discussion in Section 3).
4. Some of the BNL report conclusions should be adjusted to better reflect: a) the actual evidence gathered in the report, and b) any modification of aim and emphasis that may be put in effect as result of the current review process and incorporation of reviewers’ recommendations (Section 5).

### 3. Report Objectives

The objectives of the BNL report are stated in Section 1.1, as follows:

**“1. Develop criteria for evaluating reliability models of digital systems. These draft criteria could eventually provide input to the technical basis for risk-informed decision-making.**

**2. Review reliability models developed using traditional methods, such as fault tree and Markov methods, against the criteria to determine the capabilities and limitations of the state-of-the-art of digital system reliability models using traditional methods.**

**3. Identify traditional methods to further explore that represent a spectrum of capabilities for modeling and quantitatively assessing the reliability of digital systems.”**

Much discussion took place at the May 23-24, 2007 expert panel meeting concerning the correct interpretation of these objectives and whether the work documented in the body of the letter report consistently reflected and fulfilled them. With the benefit of that discussion one can add the following explanatory observations:

- Objective 1 is oriented towards developing criteria that may provide a basis for regulatory evaluation of analytical models of NPP digital systems that may be developed to produce risk scenarios and associated risk estimates.
- Objective 2 is oriented towards a trial application of the developed criteria to a set of pre-existing “models” that were developed by various sources, using a variety of “traditional methods.”

- Objective 3 is oriented toward down-selecting, from the initial set of traditional methods used in the various “models” evaluated and/or initially reviewed, a limited subset to further explore and evaluate for applications in the regulatory review arena.

Objective 1 is an easy-to-understand objective, which is also fully consistent with the general context and scope of the work carried out by the BNL team. No further comments are needed, except that the regulatory perspective of the model evaluation criteria is a key element of the objective that needs to be made clear to the reader for his/her correct interpretation of the objective.

Objective 2, taken at face value, also appears to be easy to understand and justify. However, when this objective is considered in combination with its companion Objective 3, several issues come forward, some of which were discussed at some length at the May 23-24 expert panel meeting:

- One issue concerns the selection of models to evaluate based on the distinction between “traditional” and “advanced” (and/or “dynamic”) modeling methods, i.e.: what is the definition of “traditional method” as opposed to “advanced dynamic method”?

This issue cannot be truly resolved in a clear-cut fashion, but one can recognize that it may be of limited importance in the context of the BNL activity and of the overall NRC research on digital systems risk and reliability. This is for two reasons:

1. The main drivers in the selection of models to evaluate with the criteria developed under Objective 1 appear in practice to have been:
    - a. the availability of a documented “real life” application of a traditional method (which the report refers to as a “model”) to a relatively large scale system, and:
    - b. the perceived relevance of the existing application to the subject of NPP digital systems risk.
  2. Advanced methods are being evaluated in a separate, but coordinated NRC research project.
- A second, and more serious issue, concerns what appears to be a logical disconnect between Objective 2 and 3, which has not yet been resolved, even after discussion at the review panel meeting. At the panel review, the report authors and sponsors clarified that the intent of the evaluation was to evaluate digital system “models” that were available and accessible, against the set of initially developed criteria, and not to apply the criteria to evaluate “methods.”

The logical disconnect occurs because the evaluation conducted under Objective 2 is de-facto translated, under Objective 3, into a down-selection of methods to further explore. Thus, although the declared intent of the report is to evaluate models, a de-facto judgment of goodness and suitability is transferred to the underlying methods, each taken as if it were a monolithic block, rather than a combination of many features matching or not the spectrum

of criteria developed under Objective 1.

The undesirable outcome that ensues is that, by following the above reasoning and course of actions, methods with potential good features and suitability for application in the regulatory context are excluded from further consideration, ostensibly because a good example of application fitting the report evaluation criteria was not readily available to the report authors.

A related outcome, perhaps even more undesirable, is that given the way the two objectives are stated and applied, outside readers will almost without doubt interpret the report evaluations to be general judgments passed on the suitability of the methods used in the various applications examined.

### **3.1 Recommendations on Statement and Application of the Report Objectives**

The good news concerning the above is that the identified issues appear to be addressable in a reasonable fashion that would not be disruptive with regard to the work already carried out by BNL:

#### **A. Distinction between “traditional” and “advanced” methods**

As mentioned above, there are some good reason not to consider this a pressing issue. One of the experts in the review panel suggested a possible definition of “traditional” method as “one that is commonly used, well established, including large-scale applications.” This reviewer’s recommendation is that a definition along those lines be used, perhaps refined to read as follows:

“A traditional method is one that has been fully demonstrated used, and established, including production-scale applications.”

This definition tempers the requirements following from the terms “commonly used” and “large-scale,” mostly in recognition of the need to keep the horizon of methods open across industries and beyond the NPP world. What is “large scale” in one industry may be not so large when viewed from another industry’s perspective. Moreover, most digital system analysis applications, even in production environments, have been at least in part exploratory and limited for one reason or another to a specific subsystem or set of subsystems.

#### **B. Evaluation of “models” vs. “methods”**

The recommendations to address this issue – in this reviewer’s opinion a serious one that should be corrected with high priority – are two-fold:

- a. Objective 2 should be restated to say that the primary purpose of the evaluation of models against criteria is test the use of the criteria against existing applications, with an accompanying objective of also better understanding the features of the methods used in these applications.

- b. Objective 3 should also be restated to say that the results of the evaluations conducted per Objective 2 are used to identify features of existing methods that can be further explored and applied within an overall PRA type of framework to model and quantitatively assess the reliability of digital systems.

Applying Recommendation a) has limited impact on the existing contents of the report, whereas Recommendation b) would require a shift in the way Objective 3 is intended to lead to follow-on activities. That is, instead of a selection of a “method” as a whole for follow-on use and evaluation, the selection would have to identify specific aspects and portions of a method that can be assembled into a PRA implementation.

The selection of what one may call a “hybrid method” (or methods) was suggested by more than one reviewer at the expert panel sessions, and in fact there is nothing novel or unusual about this approach since a typical PRA framework is indeed a hybrid model that cobbles together a number of different modeling and assessment techniques, i.e.:

- event trees (and in some cases, especially in the aerospace industry, event sequence diagrams)
- fault trees (and in some cases reliability block diagrams)
- a whole assembly of failure rate and failure-on-demand probability quantification techniques and formulations.

In summary, there is really little reason to select a method on the basis on one “model” that has been evaluated, assuming all along that such application is a good representation and illustration of all the features of the method. The two recommendations presented here substantially reduce the potential “political liabilities” that one may incur in the technical community by making such a, real or perceived, leap of judgment. This is important, given the environment and type of audience for the BNL study.

Aside from political considerations, an identification of specific method features that well match criteria, leading to the trial application of a hybrid method (or methods) using such features, appears to be the most technically sound, useful and insightful path towards practical and effective applications of digital system reliability and risk modeling methodology.

#### **4. Report Review of NASA Reliability Models**

The BNL draft letter report contains, in Appendix D, a review and evaluation of the “NASA Reliability Methods” (concerning digital systems and software). Because of his professional background and involvement, the reviewer is quite familiar with NASA PRA applications in general and with digital systems applications in particular. This specifically includes the “conditional risk model” presented and documented in the NASA PRA Procedures Guide.

This position of familiarity has made it possible for the reviewer to identify some misinterpretations

and inaccuracies in the BNL review of the NASA methodologies and applications, which it is appropriate to address here, also in relation to the recommendations made earlier in Section 3.

The main misunderstanding is relative to the concept of “conditional risk model” set forward in the NASA PRA Procedures Guide, and associated confusion with regard to the use of some specific analytical technique or other within the conditional risk model framework.

The BNL report states:

**“In the conditional risk model, hardware failure conditions are used to define the boundary conditions for modeling software failures. For each boundary condition, a software failure probability, independent of how long the software is running, is estimated using a reliability growth model. In two examples, a spacecraft attitude control system and a fluid tank control system, the method was applied. As discussed in [Chu 2006b], for control systems, software failure rate is a more appropriate parameter because the longer the control system is operating, the more likely that a triggering event would take place. Therefore, the conditional risk model of the NASA PRA procedures guide is not consistent with the framework for probabilistic modeling of software failures. It should be revised to include consideration of the duration of operation in estimating the software failure probabilities.”**

The above interpretation of the NASA model is erroneous, as essentially the framework expressed the probability of a digital system software failure as:

- an unconditional rate of occurrence per unit time or per mission (i.e., rate per unit time multiplied by mission time duration) of the system condition / triggering event,
- multiplied by the conditional probability of digital system / software failure, given the occurrence of the triggering event.

Modeling digital system software failures in the fashion set forward by the NASA PRA procedures guide is based on the actual NASA and general space system (i.e., including DoD) experience with software related failures that have led to loss of missions. In addition, modeling a risk scenario via the frequency of an “initiating event” multiplied by the conditional probability of the event or chain of events that may follow is standard PRA modeling and quantification practice.

Part of the confusion in the assessment of the NASA method may be ensuing from the NRC and BNL concern with “digital systems” failures in general, as opposed to the specific emphasis on “software related failures” in Section 11 of the NASA PRA Procedures Guide. The primary intent of Section 11 is indeed to address software-related risk modeling, which is seen by NASA as the part of digital systems modeling with which PRA practitioners are mostly unfamiliar. However, the section provides a real-life example of modeling and quantification of digital space system risk, in which some failure scenarios are driven only by the hardware portion of the digital system (i.e., sensor interface, CPU, etc.), some only by the software portion, and some by a combination of the two.

Another important point apparently missed in the BNL review is that the NASA PRA Procedures

Guide digital system modeling approach is intended to provide not a recipe for one specific combination of techniques (i.e., DFM or the Schneidewind software reliability growth model, which are used as individual elements of the application examples provided) but a flexible framework, tailorable in different types of detailed implementation, but maintaining in general the following characteristics:

- “upward” (i.e., scenario-level compatibility with the “standard” PRA event-tree / fault-tree modeling paradigm;
- ability to be “appended” and completed “downward” (i.e., in the direction of more detailed model development) with either traditional PRA modeling and quantification methods (e.g., ET / FT, Bayesian failure rate estimation, etc.) or more “advanced” or specifically software-oriented methods (DFM, Dynamic FT/Markov, SW reliability growth methods of various nature etc.).

A recently published NASA report (“Risk-Informed Safety Assurance and Probabilistic Risk Assessment of Mission-Critical Software-Intensive Systems,” AR 07-01, ASCA, June 2007) more fully illustrates and documents the NASA PRA Procedures Guide Section 11 framework.

#### **4.1 Recommendations Concerning Report Review of NASA Reliability Models**

Given the obvious sensitivity carried by evaluations and assessments of methodologies developed by other parties, and especially in this case by another U.S. Government agency, it seems appropriate to recommend a more careful review and re-examination of the substance and contents of the NASA methods and models being assessed, also in light of the more in depth and easily accessible information provided by the NASA report identified above.

The above may also apply to some of the other models and methods reviewed by BNL, but the reviewer cannot extend any recommendations in such direction since he is not as specifically familiar with these models and methods as he is with those developed by NASA.

### **5. Report Conclusions and Recommendations**

In its concluding Section 6.2, the report indirectly indicates a favorable view of “FT/ET” and Markov methods, by stating:

**“ ... The identified weaknesses of the FT/ET and Markov methods are not believed to be inherent weaknesses of these methods themselves, but rather weaknesses in the application of these methods in the studies reviewed. The FT/ET and Markov methods are very general and flexible, and it may be possible to use them to develop reasonable digital system reliability models if the identified weaknesses in the studies are addressed.”**

The above statement is probably correct with respect to the use of ET/FT modeling as a “classical” PRA top level framework, but it is less true for Markov modeling, which is not quite as flexible and presents a whole array of problems in terms of the “quantification burden” that it carries alongside.

Markov models have in fact been used in PRA occasionally, and primarily as a complement to FT techniques, but not as the “hosting” framework. If accepted for Markov modeling, the statement would also arguably be true, or truer, for other methods that were either not considered (e.g., Bayesian Belief Networks, Petri Nets) or given only a summary examination leading to not quite accurate conclusions (see Section 4 above).

The indirect suggestions concerning FT/ET (perhaps to be better referred to as ET/FT, since that is the typical logic order of PRA model development) and Markov, or any other conclusions that may be drawn concerning the selection of methods for further examination (e.g., using the “models” with the higher scores in Section 6.1.3 of the BNL report as “templates” in future research) lead back to the issues and contradictions intrinsic to the evaluation of “models” versus “methods.” This has already been addressed and discussed at some length above in Section 3.

With respect to the recommendations for areas of needed improvement in the state-of-the-art which are listed in Section 6.2 of the BNL report one can generally agree, with the following caveats:

- It is unclear if the call for the “development of methods for defining and identifying failure modes and effects of digital systems” is meant to be general, or limited to the improvement of “traditional methods;” if the latter is the case, the obvious objection is that a large portion of the research community in this area does not believe this to be possible without introducing what the BNL reports considers “advanced dynamic methods.”
- Many in the community, including this reviewer, are skeptical about the feasibility of developing generic databases for digital systems failure modes and failure rates. This is because digital system “generational” design and usage-span cycles have become shorter and shorter, so that any such database becomes obsolete in validity and applicability for the newest generation of digital hardware and software components, by the time enough data has been collected from the preceding generation.
- The above comment leads to one conclusion that this reviewer has been able to draw from his own experience with modeling and assessing digital system risk, that is, that perhaps the best hope for realistic quantification of such a risk has to rely on methods that can utilize direct test results or realistic simulation results for the systems of interest. Ref. 1 discusses how this can be done, at least for certain scenarios of especially critical concern.

## **5.1 Recommendations**

It would be inappropriate, besides being also highly logically suspect, to provide here “recommendations on recommendations.”

The only general and obvious recommendation concerning this portion of the report is that, if some of the other recommendations previously presented in this review were to be incorporated in the

final version, the general aim of the concluding sections would have to be adjusted accordingly.

For example, adjustments to the report conclusions would have to be made if the evaluation of “models” were instead presented more as a means to try out the evaluation criteria contained in Section 3 of the report, while at the same time developing experience with methods and their application models.

Similarly, adjustments would be in order if the project moved more towards the idea of trying to identify the desirable features of a framework to host hybrid combinations of models and method applications, optimized to address specific types of decisions and assessment, instead of focusing on monolithic method applications.



## **Attachment F**

### **Written Comments Provided by Reviewer C**

#### **1. Continuous Control Function**

It is mainly depend upon specific plant design nevertheless the report should take into account possibility.

Digital I&C can be used for safety related continuous control function, which may have different attributes to be modeled in a quantitative model. For instance, KSNPP's auxiliary feed water actuation signal controls injection flow according to a SG level measurement. I believe new reactor designs use digital technology to accomplish the kind of continuous control functions. Different criteria may be applicable to the function.

#### **2. Test**

Test is discussed in a criterion, modeling of fault tolerance features which is a subpart of criterion of 3.4, modeling of dependencies. Even though it is generally consider that test is a method to accomplish fault tolerance, modeling the test features in digital I&C should be handled separately with consideration of the widened test capability in digital technology. For instance, short test frequency reduce using dedicated test computer reduce failure probability of tested components, but the model should take into account the test coverage.

#### **3. Capability Evaluation**

This study evaluate whether six reviewed approaches satisfy the criteria that capture the design features of a digital system and can affect the system reliability. Besides the evaluation, it should be evaluated if the approaches are able to meet the criteria.

#### **4. Mutually Exclusive Requirement**

Report (page 31) indicates that some criteria are mutually exclusive or presents alternatives for a desirable characteristic. The study should describe the mutual exclusion and alternatives in Chapter 3 and take into account them to evaluate the six approaches in the comparison.

#### **5. Definition of Criterion**

Some criteria define specific requirements to model a certain characteristic, but the necessity of the model is depended upon a detailed design feature. Chapter 3 should define the requirement to take into account the certain design characteristic. I.e. when an approach explicitly provide a justification without modeling, the approach is considered to satisfy the criterion. For instance requirement 4.3.2 requires to model loss of HVAC and requirement. If a specific cabinet design has a temperature switch that initiate an automatic cabinet trip and/or operator actions, the necessity of HVAC modeling is negligible.

## **6. KSNPP Information**

General information for KSNPP is provided in a few places in report, but the information is inconsistent. I will provide more information regarding KSNPP later.

## Attachment G

### Written Comments Provided by Reviewer D

Reviewer D provided written comments prior to the external review panel meeting. These comments were provided as annotations to the text of a pdf version of the BNL report. The context and essence of these comments are provided below.

1. Section 2.1, “Discussion of Methods,” of BNL’s report states that the FT/ET method can quantitatively evaluate the detailed failure modes of the plant. He pointed out that an exception is when the failures are statistically dependent.
2. Section 2.1, “Discussion of Methods,” of BNL’s report states that the FT/ET method does not explicitly treat the timing of events in accident sequences, but only accounts for them in an implicit way (i.e., through the specific events included in the ETs and their order of occurrence). He pointed out that this implicit way may not be able to account for the competition between top events.
3. Section 2.1, “Discussion of Methods,” of BNL’s report states that the FT/ET method considers interactions with plant processes only implicitly in an approximate way (primarily through the system success criteria). He pointed out that in addition, non-coherence due to diagnostic and recuperative capabilities of digital I&C systems may be a limitation.
4. Criterion 1.2 states “A probabilistic model of a digital system should be modeled at least at a level of detail for which the microprocessors are separately modeled.” He pointed out that this criterion is unclear.
5. Section 3.2, “Identification of Failure Modes of the Digital System,” of BNL’s report states that ideally, the FMEA and these tools would be used in combination to identify more vulnerabilities of the system in a more reliable way than using FMEA alone. He pointed out that by design, FMEA is intended to identify immediate impacts, not to model fault propagation. So while the statement is correct, it should emphasize this intent.
6. Section 3.2, “Identification of Failure Modes of the Digital System,” of BNL’s report states that failure modes, failure causes, or failure effects are frequently mixed up, defined ambiguously, and sometimes they overlap or are even contradictory. He pointed out that these statements should be supported by citations from the literature.
7. Section 3.2, “Identification of Failure Modes of the Digital System,” of BNL’s report states that in an attempt to address the aforementioned problems with the current software failure categorization methods, a software failure categorization framework that involves definition of generic failure modes and failure causes was developed. He asked in which report this framework is presented.
8. Section 3.2, “Identification of Failure Modes of the Digital System,” of BNL’s report states

that an obvious way of defining failure modes is in terms of the functions of the system or components, e.g., an analog input module of a system may fail to convert the input signal to the correct digital signal for the system to process, and a CPU may fail to generate the correct output signals. He pointed out that this is not necessarily the case, and mentioned the example that a controller may generate arbitrary outputs or may undergo Byzantine failures.

9. Criterion 2.1 states “A technique such as FMEA should be applied at least to a level of detail corresponding to the basic components of the system, such as microprocessors.” He pointed out that this level of detail may be neither feasible nor necessary, depending on device functionality and data availability.
10. Criterion 2.2 states “Supporting analysis should be carried out to determine how specific features of a design such as communication, voting, and synchronization could affect the operation of the system. It should determine if the specific design feature could introduce dependent failures that should be modeled.” He has the same comment as the one for Criterion 2.1.
11. Criterion 2.3 states “The information associated with the probabilistic model of a digital system should provide justification that the design requirements of the digital system are unambiguous, complete and consistent, and that these requirements have been implemented in the system.” He pointed out that this criterion is unclear.
12. Section 3.3, “Modeling of Software Failures,” of BNL’s report states that hardware fails due to factors such as wear and tear, while a software failure happens due to the presence of a fault in the software and the occurrence of a specific set of input data. He asked the question “Due to specification error?”
13. Section 3.3, “Modeling of Software Failures,” of BNL’s report states that the occurrence of the input is random and can be modeled in terms of failure rates and failure probabilities. He pointed out that this statement is debatable.
14. Section 3.3, “Modeling of Software Failures,” of BNL’s report states that for a protection system, a failure rate can be used to model errors introduced during software updates. He pointed out that this statement needs substantiation by references.
15. Section 3.3, “Modeling of Software Failures,” of BNL’s report states that one way in which software failures may be explicitly included in the logic model is by somehow developing a model of behavior of the software and including it in the logic model of the rest of the NPP, and that this approach has been named “system-centric.” He does not believe a software model is necessary for the system-centric approach.
16. In the bottom paragraph of page 12 of BNL’s report, “Thereport” is a typo.
17. He suggests to rephrase Criteria 4.1.3 and 4.1.4 in a more non-device-specific manner.

18. Criterion 4.4 states that the digital systems of a plant should be examined to determine if there are dependencies due to sharing digital hardware. Such a dependency should be modeled, e.g., by linking fault trees. He pointed out that “e.g., by linking fault trees” is unclear.
19. Criterion 4.4 also states that if RPS and ESFAS are implemented using the same digital hardware, a conservative approach for accounting for this dependency is assuming that RPS is failed in those sequences with ESFAS failed due to digital failures, and vice versa. He pointed out that this statement is unclear.
20. In the discussion on “Modeling of Fault Tolerance Features,” BNL’s report states that the objective of a fault-tolerant feature is to have a positive impact on the risk metrics of a system, such as the system’s reliability. On the other hand, a fault-tolerant feature may fail to detect and/or fix a failure mode that it was designed to catch. He pointed out that if fault-tolerance relies on self-testing, the system may be vulnerable during the testing process.
21. Section 3.5, “Human Errors,” of BNL’s report states that once a digital system has been installed and is operational in a nuclear power plant (NPP), an upgrade may introduce new errors into the system. This type of failure also may happen when upgrading an analog system. However, it appears that it has a higher probability of occurring when upgrading a digital system due to the greater complexity of these systems. He pointed out that the last statement needs to be justified by a reference.
22. Under “Requirements” of the Section 3.5 “Human Errors,” the BNL’s report states that two types of human errors that are related to a digital system are the introduction of faults when upgrading its hardware or software, and poorly designed or implemented man-machine interfaces (MMI). The human reliability analysis of the probabilistic model should take into account these types of failures. Regarding the last sentence, he would simply say “The probabilistic model should take into account these types of failures.” He pointed out that there may be data available.
23. Criterion 5.2 requires modeling of human errors due to poor design of MMI. He asked if the term MMI has been defined earlier.
24. Section 3.6, “Ease of Integration with a PRA Model,” of BNL’s report states that one way to integrate a model of a digital system with an existing PRA model is by directly integrating the system model with the PRA model. Since the current PRAs use the FT/ET method, this approach can only be achieved by using a fault tree model of the digital system. He recommended to replace the words “fault tree” by “ET/FT” in the last sentence.
25. Section 3.6, “Ease of Integration with a PRA Model,” of BNL’s report states that one way to integrate a model of a digital system with an existing PRA model is by using the results from the model of a digital system in a PRA in a consistent way. This approach basically consists of developing a model of a digital system using a technique such as the Markov method. He recommended to replace the words “such as the Markov method” with “that can

account for the all the relevant features of the digital system.”<sup>4</sup>

26. Criterion 6.1 states that a fault tree model of a digital system should be easy to integrate with a PRA model. All other methods require additional efforts in integration. He recommended to replace the words “fault tree” by “ET/FT.”
27. A paragraph in the middle of page 23 of BNL’s report starts with the sentence “Thereview of the availability of hardware data for digital components reveals that ...” He pointed out that “Thereview” is a typo.
28. Point 6 in pages 32 and 76 of BNL’s report states that based on the review of the 6 studies, and previous research on PRAs of digital systems, it is still believed that the two main types of “traditional” methods, i.e., fault tree/event tree and Markov, are capable of assessing these systems. He pointed out that this statement needs substantiation in view of the criteria formulated, even if the "and" in the part "fault/tree and Markov" is meant as "in conjunction with". Neither of them will pick up all the Type I or Type II interactions. It is true that Markov/CCMT is capable of meeting all the requirements, but then Markov/CCMT is not a traditional method.
29. In Section 5.2, “Application-Specific Observations,” under the heading “Modeling of the AP 1000 using the FT/ET method,” the BNL’s report states that a strength of this modeling is that software failures were explicitly included in the logic model. He asked where did the data come from?
30. In the first paragraph of Section 6.2 “Conclusions and Recommendations,” the BNL’s report states that strengths and weaknesses have been identified for the studies reviewed. The weaknesses represent limitations of the current state of the art in modeling digital systems. The identified weaknesses of the FT/ET and Markov methods are not believed to be inherent weaknesses of these methods themselves, but rather weaknesses in the application of these methods in the studies reviewed. The FT/ET and Markov methods are very general and flexible, and it may be possible to use them to develop reasonable digital system reliability models if the identified weaknesses in the studies are addressed. He pointed out that the last sentence is debatable.
31. Section 6.2, “Conclusions and Recommendations,” of the BNL’s report identified several areas of research that would enhance the state of the art. In addition to these areas, he suggested dependencies arising from Type I and Type II interactions.

---

<sup>4</sup> Reviewer D pointed out the following publication for information on these features: J. Kirschenbaum, M. Stovsky, P. Bucci, T. Aldemir, S.A. Arndt, “Benchmark Development for Comparing Digital Instrumentation and Control System Reliability Modeling Approaches”, PSA’05, on CD-ROM, American Nuclear Society, LaGrange Park, IL (September 2005).

## **Attachment H**

### **Order for Addressing Review Criteria Categories/Subcategories**

1. Category 1. Level of Detail of the Model
2. Category 2. Identification of Failure Modes of the Components of Digital Systems
3. Category 3. Software Failures
4. Category 4. Modeling of Dependencies
  - Subcategory 4.1 Communication networks/buses
  - Subcategory 4.3 Support systems
  - Subcategory 4.4 Sharing hardware
  - Subcategory 4.5 Interactions of digital systems with other systems
  - Subcategory 4.6 Modeling of fault tolerance features
  - Subcategory 4.2 Common cause failures
5. Category 7. Probabilistic Data
  - Subcategories 7.1-7.10 Hardware failure data
  - Subcategories 7.11-7.12 Software failure data
6. Criteria 8.4-8.5. Uncertainty
7. Category 6. Ease of Integration with a PRA Model
8. Category 5. Human Errors
9. Criteria 8.1-8.3. Documentation and Results

## **APPENDIX B**

### **DETAILED FMEA OF THE DFWCS AT DIFFERENT LEVELS<sup>1</sup>**

**Appendix B.1: Top-level FMEA of the DFWCS (Page B-1)**

**Appendix B.2: FMEA at Level of the DFWCS Modules (Page B-2)**

**Appendix B.3: FMEA at Level of Major-Component-of-Module of the DFWCS (Page B-89)**

<sup>1</sup> The FMEAs presented here make extensive use of the hazard analyses performed by the plant.



**Appendix B.1: Top-level FMEA of DFWCS**

**Table B.1-1 Top-level FMEA of DFWCS**

Mode of operation of the plant: Power operation Mode of operation of the MFW: High power		
Failure Mode	Detection of Failure Mode	Failure Effect on Main Feedwater System
No or “low” signal from DFWCS to controlled components	Indications in control room of low feedwater flow and low level in steam generator(s) (SGs)	Low level in SGs causes reactor trip
		Reduction of level in SG(s) causes steam generator tube rupture (SGTR)
“High” signal from DFWCS to controlled components	Indications in control room of high feedwater flow and high level in SGs	Excessive feedwater to steam generator(s) causes reactor trip
Abnormal fluctuations of signal from DFWCS to controlled components	Depending on frequency and severity of fluctuations, operators in control room may be able to detect changes in feedwater flow and in level in SGs	Effect is expected to be similar to the one resulting from the previous two failure modes
Failure to transfer to low-power mode when reactor power decreases below 15% and remains above about 2%	Indications in control room of high level in SGs	A mismatch between the power produced by the reactor and the cooling of the SGs by the DFWCS. The mismatch may result in excessive feedwater to SGs causing a reactor trip.

**Appendix B.2 FMEA at Level of DFWCS Modules**

The next level of the FMEA includes the modules of DFWCS. The major modules of the DFWCS include Main CPU, Backup CPU, MFV Controller, BFV Controller, FWP Controller, PDI Controller, and the optical isolator that is related to the WDT signal. The FMEA is performed based on failures of input/output signals that reflect the failure modes of these modules. Thus, the FMEA of analog and digital backplanes tabulated is actually for the Main and Backup CPUs because the backplanes contain all input/output signals of the CPU.

**Table B.2-1 FMEA of Analog Backplane A (FY1111B/C1)**

FMEA of Analog Backplane A (FY1111B/C1)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Analog Backplane - Loss of communications	There is no direct indication of the loss of communication. Failure of the CPU would send an alarm to the Plant Computer.	The application software checks that there are no errors (UMAC_NO_ERROR). Loss of communication would result in an error due to no response from the analog board causing the analog inputs to be invalid which would result in a failure of the CPU. A failover to the B/U CPU will take place.	It is assumed that the loss of communication to the CPU has no effect on other functions of the ISA bus.
Analog Backplane - Loss of power (5V source)	There is no direct indication of the loss of 5V source. Failure of the CPU would send an alarm to the Plant Computer.	The application software checks that there are no errors (UMAC_NO_ERROR). Loss of 5V source would result in an error due to no response from the analog board causing the analog inputs to be invalid which would result in a failure of the CPU. A failover to the B/U CPU will take place.	

FMEA of Analog Backplane A (FY1111B/C1)

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<p>Channel 1 - Feedpump A Demand (Output) Failed Hi OOR</p>	<p>There is no direct indication of the failure. The Main CPU deviation logic will be alarmed at the Plant Computer.</p>	<p>The Main CPU deviation logic will detect a large deviation between the CPU demand and the main FWP track signal and the B/U CPU will initially track the FWP signal to the failed value. The Main CPU will then fail, transferring control to the B/U CPU. When the B/U CPU assumes control it will retrieve the pre-failure demand signal to use for its initial output. Control will be maintained by the B/U CPU and the Operator will be alerted by the DFW system trouble alarm on the Plant Computer. The Lovejoy Control system will detect a short duration increase in pump demand and will interpret this as an HIC failure. Lovejoy will then transfer control to Diagnostic Manual and maintain the FWP speed at the pre-failure speed demand.</p>	

FMEA of Analog Backplane A (FY1111B/C1)

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 1 - Feedpump A Demand (Output) Failed Low OOR	There is no direct indication of the failure. The Main CPU deviation logic will be alarmed at the Plant Computer.	The Main CPU deviation logic will detect a large deviation between the CPU demand and the FWP tracking signal and the B/U CPU will initially track the FWP signal to the failed value. The Main CPU will then fail, transferring control to the B/U CPU. When the B/U CPU assumes control, it will retrieve the pre-failure demand signal to use for its initial output. Control will be maintained by the B/U CPU and the Operator will be alerted by the DFW system trouble alarm on the Plant Computer. The Lovejoy Control system will detect a short duration increase in pump demand and will interpret this as an HIC failure. Lovejoy will then transfer control to Diagnostic Manual and maintain the SGFP speed at the pre-failure speed demand.	Need to confirm how the Lovejoy controller detects the failure.
Channel 1 - Feedpump A Demand (Output)- Excess Drift, or Step Change (in range)	There is no direct indication of this failure.	Because rate failures are not detected by the DFW system, a step change with a magnitude less than the deviation setpoint may result in a system flow transient. Because of this, the deviation setpoint should be set at a value at which the DFW control system can compensate for without resulting in a plant trip.	

FMEA of Analog Backplane A (FY1111B/C1)

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 2 - Bypass Valve Demand (Output) Failed Hi OOR	There is no direct indication of the failure.	The BFV demand signal is commanded to zero during high power mode. Should the BFV demand increase, the MFV demand will decrease as during a valve transfer, limiting the induced transient. The CPU deviation logic for the BFV demand signal is inhibited during High Power Mode Operations.	
Channel 2 - Bypass Valve Demand (Output) Failed Low OOR	This is no direct indication of the failure.	<p>The BFV demand signal is normally at zero during high power mode. If the BFV demand signal remains at zero, nothing will happen.</p> <p>This fault will remain undetected until a valve transfer occurs. At this time, the Main CPU deviation logic becomes active and will detect a large deviation between the CPU demand and the BFV signal. The B/U CPU will track the BFV signal to the failed value. The Main CPU will then fail, transferring control to the B/U CPU. When the B/U CPU assumes control, it will use the tracked demand signal for its initial output. Control will be assumed by the B/U CPU and the Operator will be alerted by the DFW system trouble alarm on the Plant Computer.</p>	

FMEA of Analog Backplane A (FY1111B/C1)

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 2 - Bypass Valve Demand (Output) Excess Drift or Step Change (in range)	There is no direct indication of the failure.	Because rate failures are not detected by the DFW system, a step change with a magnitude less than the deviation setpoint will result in a system flow transient. Because of this, the deviation setpoint should be set to a value at which the DFW control system can compensate for without resulting in a plant trip.	

FMEA of Analog Backplane A (FY1111B/C1)

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<p>Channel 3 - Main Valve Demand (Output) Failed Hi OOR</p>	<p>There is no direct indication of the failure. The Main CPU deviation logic will be alarmed at the Plant Computer.</p>	<p>The failed signal will be sent to the MFV controller causing the MFRV to open wider. The Main CPU deviation logic will detect a large deviation between the CPU demand and the MFV track signal. The B/U CPU will initially track the MFV signal to the failed value. The Main CPU will then fail, transferring control to the B/U CPU. When the B/U CPU assumes control, it will retrieve the pre-failure demand signal to use for its initial output. Control will be maintained by the B/U CPU and the operator will be alerted by the DFW trouble alarm on the Plant Computer. The FWP speed will be momentarily affected because the high auctioneered MFV signal is used to control the FWP speed. This SGFP demand transient could cause the Lovejoy Control system to interpret an HIC failure which would transfer SGFP control to Diagnostic Manual and maintain SGPR speed at the pre-failure value.</p>	

FMEA of Analog Backplane A (FY1111B/C1)

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<p>Channel 3 - Main Valve Demand (Output) Failed Low OOR</p>	<p>The PDI controller will display a “MFV Fail” message.</p> <p>A deviation message is activated by the Main CPU, after a settable, predetermined time delay. (This message may not be generated, because the PDI controller is expected to take over.)</p>	<p>The MFV controller will initially forward the failed demand signal to the MFRV positioner, PDI controller, and the CPUs of the other S/G. The PDI controller will then detect the signal failure and automatically become the manual controller for the MFRV using the old value in the circular buffer. The MFRV must be manually controlled from the PDI controller.</p> <p>The failed signal will be sent to the CPUs of other SG, and probably will not affect the FWP speed calculation.</p>	<p>The response specified in plant analysis probably will not take place, because the PDI controller has a scan time of not exceeding 100 milliseconds, while the CPU failover logic has a 1 second delay.</p> <p>The MFV demand signal is also sent to the CPUs of the other S/G and used in the FWP speed calculation.</p>
<p>Channel 3 - Main Valve Demand (Output) Excess Drift, or Step Change (in range)</p>	<p>There is no direct indication of the failure.</p>	<p>Because rate failures of the BFV demand signal are not detected by the DFW system, a step change with a magnitude less than the deviation setpoint will result in a system flow transient. Because of this, the deviation setpoint should be set at a value at which the DFW control system can compensate for without resulting in a plant trip.</p>	
<p>Channel 4 - S/G 12 FW Temp (Input) OOR</p>	<p>A deviation alarm will be sent from the Main CPU to the Plant Computer.</p>	<p>The OOR condition and deviation will be detected by the Main CPU. The signal becomes invalid and the other signal is used. There is no effect on control. The signal is only used during low power operation.</p>	
<p>Channel 4 - S/G 12 FW Temp (Input) Excess Drift or Step Change (in range)</p>	<p>If the deviation is large enough, an alarm will be sent from the Main CPU to the Plant Computer.</p>	<p>The temperature signals are averaged and a deviation will not significantly affect low power control.</p>	



FMEA of Analog Backplane A (FY1111B/C1)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 5 - S/G 11 FW Temp (Input) OOR	A deviation alarm will be sent from the Main CPU to the Plant Computer.	The OOR condition and deviation will be detected by the Main CPU. The signal becomes invalid and the other signal is used. There is no effect on control. The signal is only used during low power operation.	
Channel 5 - S/G 11 FW Temp (Input) Excess Drift or Step Change (in range)	When the deviation is large enough, an alarm will be sent from the Main CPU to the Plant Computer.	The temperature signals are averaged and a deviation will not significantly affect low power control.	
Channel 6 - S/G 11 Feedpump A Bias (Input) Fails High or Low OOR	<p>A deviation alarm will be sent to the Plant Computer from the Main CPU. The FWP controller will activate a local alarm when the Main CPU demand signal differs from the B/U CPU demand signal by an amount exceeding a setpoint.</p> <p>The FWP controller will activate a local deviation alarm when the Main CPU demand signal differs from the B/U CPU demand signal by an amount exceeding a setpoint.</p>	The OOR condition will be detected by the Main CPU, and a deviation alarm will be sent to the Plant Computer. The Main CPU will send the pump demand calculated with the failed bias signal to the FWP controller.	<p>The effects depend on whether or not the failed signal at the Main CPU would also be received by the BFV controller.</p> <p>It is assumed the Main CPU will send the pump demand calculated with the failed bias signal to the FWP controller.</p>
Channel 7 - S/G 12 Main Valve Tracking (Input) Failed Hi OOR -	There is no direct indication of the failure. The increase in pump speed could be alarmed using the future D/P signals.	This is one of two signals (S/G 12 and S/G 11) used for high select to determine the FWP speed. A failed high signal will increase the pump speed which would cause a controllable disturbance at high power but could result in loss of control at low power. The Lovejoy Control System may detect this large change and transfer to Diagnostic Manual mode.	

FMEA of Analog Backplane A (FY1111B/C1)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 7 - S/G 12 Main Valve Tracking (Input) Failed Low OOR	There in no direct indication of the failure.	This is one of two signals used for high select to determine the FWP speed. A failed low signal will have minimal effect on system operation.	
Channel 7 - S/G 12 Main Valve Tracking (Input) Excess Drift, or Step Change (in range)	There in no direct indication of the failure.	A decrease in value would have no effect; an increase in value would have little effect and be compensated for by an adjustment in the controlling valve.	
Channel 8 - S/G 11 FWP A Tracking (Input) Fail High OOR, Fail Low OOR, and Excess Drift, or Step Change (in range)	There in no direct indication of the failure. The Main CPU failure will result in an alarm being sent to the Plant Computer.	The deviation between the CPU and controller will cause a failover to the B/U CPU as long as the deviation setpoint is exceeded. If the deviation setpoint is not exceeded, control will continue as the controlling demand signal is valid.	
Channels 9-12 are spares			

FMEA of Analog Backplane A (FY1111B/C1)

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<p>Channel 13 - MFRV LVDT #2 Fail High or Fail Low</p>	<p>A MFV Large Deviation alarm will be activated on the plasma display unit and the associated CPU deviation annunciator will activate, if the deviation between the two LVDT inputs exceeds the MFV_Deviation setpoint. If the Diagnostic Transfer mode is enabled, then it will transfer to Lockout.</p>	<p>If the deviation between the two LVDT inputs exceeds the MFV_DEVIATION setpoint, the Diagnostic Transfer mode will transfer to Lockout.</p> <p>If the MFV_DEVIATION setpoint is not exceeded and the MFV_DEADBAND setpoint is exceeded by the Demand-LVDT deviation, where the LVDT is the average of the two LVDT signals, the DMD-LVDT deviation will be accumulated over the subsequent cycles. If the accumulation exceeded the MFV_ACCUMULATION setpoint, and the Diagnostic Transfer control mode is ENABLED, the opposite positioner will be put into service and the control mode will be shifted to LOCKOUT.</p>	<p>It is not known what the CPU deviation annunciator is. It probably is a local annunciator on the PDU. The CPUs do not have direct connection to the control room annunciators.</p>
<p>Channel 14 - MFRV LVDT #1 Fail High or Fail Low</p>	<p>Same as above.</p>	<p>Same as above.</p>	<p>Same as above.</p>

FMEA of Analog Backplane A (FY1111B/C1)

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<p>Channel 15 or 16 - MFRV Differential Pressure #2 or #1 Fail High</p>	<p>The incorrect gooseneck flow and accumulated volume will be displayed on the Analog Inputs display page of the PDU.</p>	<p>The failed high signal will falsely indicate a larger than normal differential pressure, which would result in an incorrectly high accumulated gooseneck volume and prevent needed gooseneck purge. It is not clear what adverse effects would result when the needed purge is not performed.</p>	<p>The gooseneck is an upward bend and loop installed down stream of the feedwater nozzle of replacement steam generators to prevent flow of steam generator fluid upstream. When a gooseneck purge is needed, as determined by the accumulated gooseneck volume being less than the minimum volume setpoint, the BFRV alarm status becomes GSNECK PURGE and the associated CPU deviation annunciator will activate. The operator has to manually purge the Gooseneck. It is not known what the CPU deviation annunciator is. It probably is a local annunciator on the PDU. The CPUs do not have direct connection to the control room annunciators.</p>

FMEA of Analog Backplane A (FY1111B/C1)

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<p>Channel 15 or 16 - FRV Differential Pressure #2 or #1 Fail Low</p>	<p>The incorrect gooseneck flow and accumulated volume will be displayed on the Analog Inputs display page of the PDU.</p>	<p>The failed low signal will falsely indicate a smaller than normal differential pressure, which would result in an incorrectly low accumulated gooseneck volume and premature gooseneck purge. It is not clear what adverse effect would result when the premature purge is performed.</p>	<p>The gooseneck is an upward bend and loop installed down stream of the feedwater nozzle of replacement steam generators to prevent flow of steam generator fluid upstream. When a goose-neck purge is needed, as determined by the accumulated gooseneck volume being less than the minimum volume setpoint, the BFRV alarm status becomes GSNECK PURGE and the associated CPU deviation annunciator will activate.</p> <p>It is not known what the CPU deviation annunciator is. It probably is a local annunciator on the PDU. The CPUs do not have direct connection to the control room annunciators.</p>

**Table B.2-2 FMEA of Analog Backplane B**

According to plant information, Channels 1 and 2 are reserved for test point output. It is assumed that these are for off-line test of the digital control system. Channels 3 - 5 are spared. The failures of these analog signals are not considered here.

FMEA of Analog Backplane B			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Analog Backplane - Loss of Communications	Same as Analog Backplane A.	Same as Analog Backplane A.	Same as Analog Backplane A.
Analog Backplane - Loss of Power	Same as Analog Backplane A.	Same as Analog Backplane A.	Same as Analog Backplane A.
Channel 6 - S/G 11 Level #1: Failed Hi OOR, Failed Low OOR, and Rate	Main CPU (the controlling CPU) Fail alarm status will be displayed on PDU if the failover occurs.	The other level input #2 is used and control continues. After a delay, if the B/U CPU is healthy, a failover to the B/U CPU will occur.	Channel 6 is an input signal.  Failure effects are the same for both high and low power control modes.
Channel 6 - S/G 11 Level #1: Excess Drift or Step Change (the change is within the range)	A deviation alarm status will be actuated in the plant computer. Main CPU Fail alarm status will also be displayed if failover occurs.	A deviation between S/G 11 Level #1 signal and S/G 11 Level #2 signal will occur. A small deviation will result in a deviation alarm to the plant computer. If the deviation continues, a large deviation will result and after some delay, a failover to the B/U CPU will occur if it is healthy. Otherwise, the control will continue with the average of the two level inputs.	Channel 6 is an input signal.  Failure effects are the same for both high and low power control modes.
Channel 7 - S/G 11 Level #2: Failed Hi OOR, Failed Low OOR, and Rate	Main CPU (the controlling CPU) Fail alarm status will be displayed on PDU if the failover occurs	The other level input #1 is used and control continues. After a delay, if the B/U CPU is healthy, a failover to the B/U CPU will occur.	Channel 7 is an input signal.  Failure effects are the same for both high and low power control modes.

FMEA of Analog Backplane B			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 7 - S/G 11 Level #2: Excess Drift, or Step Change (the change is within the range)	A deviation alarm status will be actuated in the plant computer. Main CPU Fail alarm status will also be displayed if failover occurs.	A deviation between S/G 11 Level #1 signal and S/G 11 Level #2 signal will occur. A small deviation will result in a deviation alarm to the plant computer. If the deviation continues, a large deviation will result and after some delay, a failover to the B/U CPU will occur if it is healthy. Otherwise, the control will continue with the average of the two level inputs.	Channel 7 is an input signal.  Failure effects are the same for both high and low power control modes.
Channel 8 - S/G 11 FW Flow #1: Failed Hi OOR, Failed Low OOR, and Rate	Main CPU (the controlling CPU) Fail alarm status will be displayed on PDU if the failover occurs.	The other FW flow input #2 is used and control continues. After a delay, if the B/U CPU is healthy, a failover to the B/U CPU will occur.	Channel 8 is an input signal.  Failure effects are the same for both high and low power control modes.
Channel 8 - S/G 11 FW Flow #1: Excess Drift, or Step Change (the change is within the range)	A deviation alarm will be actuated in the plant computer. Main CPU Fail alarm will also be displayed if the failover occurs.	A deviation between S/G 11 FW Flow #1 signal and S/G 11 FW Flow #2 signal will occur. A small deviation will result in a deviation alarm to the plant computer. If the deviation continues, a large deviation will result. A large deviation will result in single element mode control.	Channel 8 is an input signal.  At low power control mode, a large deviation will result in inhibiting a low to high power transfer and an inhibiting transfer alarm will be displayed in the plant computer.
Channel 9 - S/G 11 FW Flow #2: Failed Hi OOR, Failed Low OOR, and Rate	Main CPU (the controlling CPU) Fail alarm status will be displayed on PDU if the failover occurs.	The other FW flow input #1 is used and control continues. After a delay, if the B/U CPU is healthy, a failover to the B/U CPU will occur.	Channel 9 is an input signal.  Failure effects are the same for both high and low power control modes.

FMEA of Analog Backplane B			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 9 - S/G 11 FW Flow #2: Excess Drift, or Step Change (the change is within the range)	A deviation alarm status will be displayed in the plant computer. Main CPU Fail alarm status will also be displayed if failover occurs.	A deviation between S/G 11 FW Flow #1 signal and S/G 11 FW Flow #2 signal will occur. A small deviation will result in a deviation alarm to the plant computer. If the deviation continues, a large deviation will result. A large deviation will result in single element mode control.	Channel 9 is an input signal.  At low power control mode, a large deviation will result in inhibiting a low to high power transfer and an inhibiting transfer alarm will be displayed in the plant computer.
Channel 10 - S/G 11 Main Steam Flow: Failed Hi OOR, Failed Low OOR, and Rate	Main CPU (the controlling CPU) Fail alarm status will be displayed on PDU if the failover occurs.	The other steam flow input is used and control continues. After a delay, if the B/U CPU is healthy, a failover to the B/U CPU will occur.	Channel 10 is an input signal.  Failure effects are the same for both high and low power control modes.
Channel 10 - S/G 11 Main Steam Flow: Excess Drift, or Step Change (the change is within the range)	A deviation alarm status will be displayed in the plant computer. Main CPU Fail alarm status will also be displayed if failover occurs.	A deviation between S/G 11 Main Steam Flow and S/G 12 Main Steam Flow signals will occur. A small deviation will result in a deviation alarm to the plant computer. If the deviation continues, a large deviation will result. A large deviation will result in single element model control.	Channel 10 is an input signal.  At low power control mode, a large deviation will result in inhibiting a low to high power transfer and an inhibiting transfer alarm will be displayed in the plant computer. Note the steam flow small deviation alarms and messages are disabled when reactor power is below the low to high power mode transfer setpoint.
Channel 11 – S/G 12 Main Steam Flow: Failed Hi OOR, Failed Low OOR, and Rate	Main CPU (the controlling CPU) Fail alarm status will be displayed on PDU if the failover occurs.	The other steam flow input is used and control continues. After a delay, if the B/U CPU is healthy, a failover to the B/U CPU will occur.	Channel 11 is an input signal.  Failure effects are the same for both high and low power control modes.



FMEA of Analog Backplane B			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 11 - S/G 12 Main Steam Flow: Excess Drift, or Step Change (the change is within the range)	A deviation alarm status will be displayed in the plant computer. Main CPU Fail alarm status will also be displayed if failover occurs.	A deviation between S/G 11 Main Steam Flow and S/G 12 Main Steam Flow signals will occur. A small deviation will result in a deviation alarm to the plant computer. If the deviation continues, a large deviation will result. A large deviation will result in single element model control.	Channel 11 is an input signal.  At low power control mode, a large deviation will result in inhibiting a low to high power transfer and an inhibiting transfer alarm will be displayed in the plant computer. Note the steam flow small deviation alarms and messages are disabled when reactor power is below the low to high power mode transfer setpoint.
Channel 12 - Neutron Flux #1: Failed Hi OOR, Failed Low OOR	A deviation alarm status may be displayed in the plant computer.	The other flux input (#2) is used and control continues.	Channel 12 is an input signal.  No deviation logic for CPU failover for neutron flux inputs.
Channel 12 - Neutron Flux #1: Excess Drift, or Step Change (the change is within the range)	A deviation alarm and an inhibit transfer alarm will be displayed in the plant computer.	Valve transfers are inhibited and control continues as long as the other flux input (#2) is valid.	Channel 12 is an input signal.  At low power control mode, the last valid flux signal is frozen to minimize any disturbance.
Channel 13 - Neutron Flux #2: Failed Hi OOR, Failed Low OOR	A deviation alarm status may be displayed in the plant computer.	The other flux input (#1) is used and control continues.	Channel 13 is an input signal.  No deviation logic for CPU failover for neutron flux inputs.
Channel 13 - Neutron Flux #2: Excess Drift, or Step Change (the change is within the range)	A deviation alarm and an inhibit transfer alarm will be displayed in the plant computer.	Valve transfers are inhibited and control continues as long as the other flux input (#1) is valid.	Channel 13 is an input signal.  At low power control mode, the last valid flux signal is frozen to minimize any disturbance.

FMEA of Analog Backplane B			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 14 - S/G 11 Level Setpoint: Failed Hi OOR, Failed Low OOR, and Excess Drift, or Step Change (the change is within the range)	A deviation alarm will be displayed in the plant computer.	A deviation between this input and the program value of the setpoint will occur. If the deviation is less than a fixed value (LEV_SPT), the control continues. Otherwise, an internal level setpoint will be used as the substitute.	Channel 14 is an input signal.
Channel 15 - S/G 11 BFRV Tracking: Failed Hi OOR, Failed Low OOR, and Excess Drift, or Step Change (the change is within the range)	No alarms are generated.	Control will continue and the CPU or BFV still sends demand output that will close the BFRV.	Channel 15 is an input signal.  At low power control mode, the deviation between the CPU and controller will cause a failover to the B/U CPU some time (deviation time delay) after the deviation setpoint is exceeded.
Channel 16 - S/G 11 MFRV Tracking: Failed Hi OOR, Failed Low OOR, and Excess Drift, or Step Change (the change is within the range)	A deviation alarm will be displayed on PDU. Main CPU (the controlling CPU) Fail alarm will also be displayed on PDU if the failover occurs.	The deviation between the CPU output and controller output will cause a failover to the B/U CPU as long as the deviation setpoint is exceeded after some time delay. If the deviation setpoint is not exceeded, control will continue as the controlling demand signal is valid.  For the Fail Hi OOR signal, the FWP speed will momentarily increase due to high MFV actioneering, which is used to compute the FWP pump demand.	Channel 16 is an input signal.

**Table B.2-3 FMEA of Digital Backplane (I/O) of Main CPU**

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of Communications</u>			
Loss of communications	The three controllers (Main, Bypass and FW Pump) alarm lights energize. The BFV controller transmits to the plant computer that one CPU has failed. The PDU shows that the Main microprocessor is failed.	Loss of communications would result in the digital signals maintaining their existing state which would cause a watchdog failure which would result in a failover to the B/U CPU.	

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of Power</u>			
Loss of Power	The three controllers (Main, Bypass and FW Pump) alarm lights energize. The BFV controller transmits to the plant computer that one CPU has failed. The PDU shows that the Main microprocessor is failed.	Loss of power would result in the digital signals changing to their unpowered state which would cause a watchdog failure which would result in a failover to the B/U CPU.	
<u>Digital Outputs</u>			
Channel 0 - Watchdog Timer (Output) fails as is	There is no direct indication of this failure. Indirect indications are annunciations of failure of the Main CPU in this CPU's PDU, and in the plant computer (from the BFV).	A failure of this output to change state would result in a Main CPU failover to the B/U CPU.	Output state: toggling (not failed). Watchdog Timer failing as is indicates that the timer identified a failure of the Main CPU.
Channel 1 - Unuseable	Not applicable (NA)	NA	Need to confirm why this channel is unuseable and whether it could have some failure mode.
Channel 2 - Power Fail (Output) fails as is	There is no indication of this failure.	A failure of this output would have no effect on control until some other condition caused the Main CPU to fail, at which time automatic control would be lost. The severity of the loss of control would depend on the CPU fault.	This channel indicates power failure or microprocessor not controlling. Output state: not energized (OK). Power Fail failing as is indicates that the Main CPU is OK.
Channel 2 - Power Fail (Output) fails to opposite state	There is no direct indication of this failure. Indirect indications are annunciations of failure of the Main CPU in the PDU of the DFWCS and in the plant computer (from the BFV).	A failure of this output would result in a CPU failover.	This channel indicates power failure or microprocessor not controlling. Output state: not energized (OK). Power Fail failing to opposite state indicates that the Main CPU is failed.

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 3 - Unuseable	NA	NA	An analysis of this channel was not found in plant information. Need to confirm why this channel is unuseable and whether it could have some failure mode.
Channel 4 - High Power Indication (Output) fails closed	There is no indication of this failure.	There is indication that the DFWCS is in high-power mode. Operation of DFWCS is unaffected, but it may be puzzling to the operators that the DFWCS remains in high-power mode even if the plant is operating in conditions corresponding to low-power mode.	Output state: energized (closed = high power). High Power Indication failing closed indicates that the DFWCS is in high-power mode.
Channel 4 - High Power Indication (Output) fails open	There is no indication of this failure.	Operation of DFWCS is unaffected, but it may be puzzling to the operators that there is no indication that the DFWCS is in high-power mode when the plant is operating in conditions corresponding to this mode.	Output state: energized (closed = high power). High Power Indication failing open does not give indication that the DFWCS is in high-power mode.
Channel 5 - Transfer Indication (Output) fails closed	There is no indication of this failure.	There is indication that the DFWCS is transferring between power modes. Operation of DFWCS is unaffected, but it may be puzzling to the operators that the DFWCS remains in a transferring state.	Output state: energized (closed = transferring). Transfer Indication failing closed indicates that the DFWCS is transferring between power modes.
Channel 5 - Transfer Indication (Output) fails open	There is no indication of this failure.	Operation of DFWCS is unaffected, but it may be puzzling to the operators that there is no indication that the DFWCS is transferring when a transfer is taking place.	Output state: energized (closed = transferring). Transfer Indication failing open does not give indication that the DFWCS is transferring between power modes.

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 6 - Low Power Indication (Output) fails closed	There is no indication of this failure.	There is indication that the DFWCS is in low-power mode. Operation of DFWCS is unaffected, but it may be puzzling to the operators that the DFWCS remains in low-power mode even if the plant is operating in conditions corresponding to high-power mode.	Output state: energized (closed = low power). Low Power Indication failing closed indicates that the DFWCS is in low-power mode.
Channel 6 - Low Power Indication (Output) fails open	There is no indication of this failure.	Operation of DFWCS is unaffected, but it may be puzzling to the operators that there is no indication that the DFWCS is in low-power mode when the plant is operating in conditions corresponding to this mode.	Output state: energized (closed = low power). Low Power Indication failing open does not give indication that the DFWCS is in low-power mode.
Channel 7 - Bypass Override Indication (Output) fails closed	There is no indication of this failure.	There is indication that the DFWCS is in Bypass Override (BPO) mode. Operation of DFWCS is unaffected, but it may be puzzling to the operators that the DFWCS is in BPO mode when they have not set the DFWCS to operate in this mode.	Output state: energized (closed = BPO mode). Bypass Override Indication failing closed indicates that the DFWCS is in BPO mode.
Channel 7 - Bypass Override Indication (Output) fails open	There is no indication of this failure.	Operation of DFWCS is unaffected, but it may be puzzling to the operators that there is no indication that the DFWCS is in BPO mode when the DFWCS has been set to operate in this mode.	Output state: energized (closed = BPO mode). Bypass Override Indication failing open does not give indication that the DFWCS is in BPO mode.
Channel 8 - Deviation Alarm (Output) fails closed	There is no indication of this failure. However, there is indication in the plant computer that the Main CPU detected a deviation.	Operation of DFWCS is unaffected.	Output state: energized (closed = deviation). Deviation Alarm failing closed indicates that the Main CPU detected a deviation.

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 8 - Deviation Alarm (Output) fails open	There is no indication of this failure.	If the Main CPU detected a deviation, there is no indication of the detection, but there is a failover to the B/U CPU. If the Main CPU did not detect a deviation, the Main CPU remains in control. In either case, operation of DFWCS is unaffected.	Output state: energized (closed = deviation). Deviation Alarm failing open does not give indication that the Main CPU detected a deviation.
Channel 9 - Transfer Inhibit (Output) fails closed	There is no indication of this failure. However, there is indication in the plant computer that the transfer of power modes is inhibited.	Operation of DFWCS is unaffected.	Output state: energized (closed = transfer inhibited). Transfer Inhibit failing closed indicates that the transfer of power modes is inhibited.
Channel 9 - Transfer Inhibit (Output) fails open	There is no indication of this failure.	If the transfer of power modes is not inhibited, there is no need for indication that the transfer is inhibited. If the transfer of power modes is inhibited, there is no indication in the plant computer that the transfer is inhibited. However, there would be indication that the transfer is inhibited in the PDU. In either case, operation of DFWCS is unaffected.	Output state: energized (closed = transfer inhibited). Transfer Inhibit failing open does not give indication that the transfer of power modes is inhibited.
Channel 10 - Spare Output	Not known	Not known	An analysis of this channel was not found in plant information. Need to confirm whether this channel could have some failure mode.

FMEA of Digital Backplane (I/O) of Main CPU

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<p>Channel 11 - Positioner Selected (Output) fails closed (fails as is)</p>	<p>There is no indication of this failure.</p>	<p>The signal from the Main CPU will be that the active positioner is A. If the accumulated deviation between the demand from the Main CPU and the position of the MFRV exceeds a setpoint value, the Main CPU will try to put into service the opposite positioner (B). However, the signal from the Main CPU will remain that the active positioner is A. If the accumulated deviation exceeded a setpoint, the positioner A may not be working properly; in this case the Main CPU will not be able to control the MFRV correctly. The impact of this loss of control of the MFRV can vary from a slight deviation of the position of the valve (with respect to the demand from the Main CPU) to the valve fully closing, leading to a reactor trip.</p>	<p>Output state: not energized (we assumed open = B positioner selected). Positioner Selected failing closed indicates that the A positioner is selected as the active positioner. An analysis of this channel was not found in plant information.</p>



FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 11 - Positioner Selected (Output) fails open (fails to opposite state)	There is no direct indication of this failure. An indirect indication is that the PDU will show that the active positioner changed from A to B.	The signal from the Main CPU will be that the active positioner is B. If the accumulated deviation between the demand from the Main CPU and the position of the MFRV exceeds a setpoint value, the Main CPU will try to put into service the opposite positioner (A). However, the signal from the Main CPU will remain that the active positioner is B. If the accumulated deviation exceeded a setpoint, the positioner B may not be working properly; in this case the Main CPU will not be able to control the MFRV correctly. The impact of this loss of control of the MFRV can vary from a slight deviation of the position of the valve (with respect to the demand from the Main CPU) to the valve fully closing, leading to a reactor trip.	Output state: not energized (we assumed open = B positioner selected). Positioner Selected failing open indicates that the B positioner is selected as the active positioner. An analysis of this channel was not found in plant information.
Channel 12 - No Failures in Microprocessor (Output) fails closed	There is no indication of this failure. However, status of the Main CPU will change to failed in PDU.	The Main CPU remains in control. On the other hand, the B/U CPU will receive signal from the Main CPU that the Main CPU failed, but it is expected that the B/U CPU will receive from the MFV the correct status (OK) of the Main CPU. We do not know how the B/U CPU handles this contradictory information.	Output state: not energized (we assumed closed = failed). No Failures in Microprocessor failing closed indicates that the Main CPU failed.  This channel is named "Deviation Alarm Status of Other CPU" in plant information.

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 12 - No Failures in Microprocessor (Output) fails open	There is no indication of this failure. If the Main CPU failed, it would be annunciated by the PDU and the plant computer.	<p>If the Main CPU is OK, operation of DFWCS is unaffected.</p> <p>If the Main CPU fails, the B/U CPU will receive signal from the Main CPU that the Main CPU is OK, but it is expected that the B/U CPU will receive from the MFV the correct status (failed) of the Main CPU. We do not know at this time how the B/U CPU handles this contradictory information.</p>	<p>Output state: not energized (<i>we assumed closed = failed</i>). No Failures in Microprocessor failing open indicates that the Main CPU is OK.</p> <p>This channel is named “Deviation Alarm Status of Other CPU” in plant information.</p>
Channel 13 - No Deviations (from Main CPU to B/U CPU) (Output) fails closed	There is no indication of this failure. However, status of the Main CPU will change to failed in PDU.	<p>The Main CPU remains in control. The incorrect signal may negatively influence the deviation decisions carried out by the B/U CPU.</p> <p>On the other hand, the B/U CPU will receive signal from Main CPU that the Main CPU failed, but it is expected that the B/U CPU will receive from the MFV the correct status (OK) of the Main CPU. We do not know at this time how the B/U CPU handles this contradictory information.</p>	<p>Output state: not energized (<i>we assumed closed = failed</i>). No Deviations from Main CPU to B/U CPU failing closed indicates that the Main CPU failed.</p> <p>This channel is named “CPU Failure Status to Other CPU” and stated unused in plant document.</p>

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 13 - No Deviations (from Main CPU to B/U CPU) (Output) fails open	There is no indication of this failure. If the Main CPU had deviations, it would be annunciated by the PDU and the plant computer.	<p>If the Main CPU is OK, operation of DFWCS is unaffected.</p> <p>If the Main CPU had deviations, the B/U CPU will receive signal from the Main CPU that the Main CPU is OK, but it is expected that the B/U CPU will receive from the MFV the correct status (failed) of the Main CPU. We do not know at this time how the B/U CPU handles this contradictory information.</p>	<p>Output state: not energized (we assumed closed = failed). No Deviations from Main CPU to B/U CPU failing open indicates that the Main CPU is OK.</p> <p>This channel is named "CPU Failure Status to Other CPU" and stated unused in plant document.</p>
Channel 14 - CPU Level Status to Other CPU (Output) fails closed	There is no indication of this failure.	The Main CPU sends a signal to the B/U CPU indicating that both S/G level signals are invalid. If the Main CPU is OK, operation of DFWCS is unaffected.	<p>Output state: not energized (we assumed closed = invalid). CPU Level Status to Other CPU failing closed indicates that both S/G level signals from the Main CPU are invalid.</p> <p>This channel is stated unused in plant document.</p>
Channel 14 - CPU Level Status to Other CPU (Output) fails open	There is no indication of this failure.	The Main CPU sends a signal to the B/U CPU indicating that both S/G level signals are valid. If the Main CPU is OK, operation of DFWCS is unaffected.	<p>Output state: not energized (we assumed closed = invalid). CPU Level Status to Other CPU failing open indicates that both S/G level signals from the Main CPU are valid.</p> <p>This channel is stated unused in plant document.</p>

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 15 - CPU Feedflow/Steamflow Status to Other CPU (Output)	NA	NA	It appears that this channel is connected to the other CPU, though it is not known how the information transmitted through this channel is used by the other CPU.  This channel is stated unused in plant document.
<u>Digital Inputs</u>			
Channel 16 - A/M Status BFV (Input) fails closed (fails as is)	There is no indication of this failure.	Operation of DFWCS is unaffected.	Input state: open circuit (closed = auto). A/M Status BFV failing closed indicates that the BFV is in auto.
Channel 16 - A/M Status BFV (Input) fails open (fails to opposite state)	There is no indication of this failure. However, there would be a discrepancy between the A/M status shown in the PDU and the one shown by the BFV.	The Main CPU would think the A/M status was manual and track instead of control. The valve demand could slowly windup and drift open. Level would be maintained as the main valve compensates for level errors.	Input state: open circuit (closed = auto). A/M Status BFV failing open indicates that the BFV is in manual.
Channel 17 - A/M Status MFV (Input) fails closed (fails as is)	There is no indication of this failure.	Operation of DFWCS is unaffected. However, if the MFV is in manual mode, the Main CPU would “think” that it is controlling the MFRV, but actually would not be controlling it. If the deviation is large enough between the Main CPU’s demand and the MFV’s demand, the Main CPU would fail, and the B/U CPU would recognize that the MFV is in manual.	Input state: open circuit (closed = auto). A/M Status MFV failing closed indicates that the MFV is in auto.

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 17 - A/M Status MFV (Input) fails open (fails to opposite state)	There is no indication of this failure. However, there would be a discrepancy between the A/M status shown in the PDU and the one shown by the MFV.	The Main CPU would think the A/M status was manual and track instead of control. The MFRV would not be controlled, so the S/G level would drift from setpoint. Operators can take manual control based on indications of incorrect S/G level and the discrepancy between the A/M status shown in the PDU and the one shown by the MFV.	Input state: open circuit (closed = auto). A/M Status MFV failing open indicates that the MFV is in manual.
Channel 18 - A/M Status FWP (Input) fails closed (fails as is)	There is no indication of this failure.	Operation of DFWCS is unaffected. However, if the FWP is in manual mode, the Main CPU would “think” that it is controlling the FWP, but actually would not be controlling it. If the deviation is large enough between the Main CPU’s demand and the FWP controller’s demand, the Main CPU would fail, and the B/U CPU would recognize that the FWP is in manual.	Input state: open circuit (closed = auto). A/M Status FWP failing closed indicates that the FWP is in auto.
Channel 18 - A/M Status FWP (Input) fails open (fails to opposite state)	There is no indication of this failure. However, there would be a discrepancy between the A/M status shown in the PDU and the one shown by the FWP.	The CPU would think the A/M status was manual and track instead of control. Pump demand would windup. The main valve would compensate for the pump speed change, giving the operators time to take control as S/G level changed when the main valve could no longer compensate. Operators can take manual control based on indications of incorrect S/G level and the discrepancy between the A/M status shown in the PDU and the one shown by the FWP.	Input state: open circuit (closed = auto). A/M Status FWP failing open indicates that the FWP is in manual.

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 19 - Reactor Trip (Input) fails closed (fails as is)	There is no indication of this failure.	No effect on DFWCS control. The DFWCS would not be able to detect a reactor trip. If a reactor trip were to subsequently occur, then the DFWCS would not ramp the MFRV shut or run back the FWP demand to minimum speed. The BFRV would open to its post trip position as determined by the feedwater bypass trip set control (1-FC-1211, 1221). The MFRV would shut after the time delay positioning relay times out. When this occurs, the Main CPU will fail due to MFRV deviation. The B/U CPU will take over the automatic control of the DFWCS in low-power mode. The associated FWP demand signal will run back to minimum speed.	Input state: open circuit (closed = not tripped). Reactor Trip failing closed indicates that there is no reactor trip.
Channel 19 - Reactor Trip (Input) fails open (fails to opposite state)	A reactor trip will occur.	<p>During a programmable validation period, no alarm messages will be actuated if a non-validated trip signal is present. After this period, a Reactor Power Large Deviation alarm will be activated on the Vuepoint alarm display alarm and event log entry will result in activation of all trip functions.</p> <p>If there is a concurrent invalid Reactor Power Input, control would be lost. The CPU would erroneously ramp the main valve shut. This failure would result in a reactor trip.</p>	Input state: open circuit (closed = not tripped). Reactor Trip failing open indicates that there is a reactor trip.

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 20 - Main / B/U CPU Identification (Input) fails closed (fails as is)	There is no indication of this failure.	Plant analysis states that “The Main CPU has no external field connections to fail.” It appears that plant analysis concludes that this failure mode cannot occur for the Main CPU. We do not have enough information to assess whether this conclusion is correct. The B/U CPU digital input is grounded. If the external connection were to fail, the B/U CPU would think it was the Main CPU and start to control versus track. As the Main CPU is selected first by the DFWCS controllers, the DFWCS would continue to operate normally. However, the B/U CPU would windup its outputs causing the B/U CPU to fail due to a deviation between the demand and controller output.	Input state: open circuit (closed = main). Main / B/U CPU Identification failing closed indicates that the CPU is the Main CPU.
Channel 20 - Main / B/U CPU Identification (Input) fails open (fails to opposite state)	There is no indication of this failure.	Plant analysis states that “The Main CPU has no external field connections to fail.” It appears that plant analysis concludes that this failure mode cannot occur for the Main CPU. We do not have enough information to assess whether this conclusion is correct. The B/U CPU is unaffected.	Input state: open circuit (closed = main). Main / B/U CPU Identification failing open indicates that the CPU is the B/U CPU.
Channel 21 - Turbine Trip (Input) fails closed (fails as is)	There is no indication of this failure.	No effect on DFWCS control. The DFWCS would not be able to detect a turbine trip. If a turbine trip were to subsequently occur, a reactor trip would follow, and the DFWCS would remain in automatic control.	Input state: open circuit (closed = not tripped). Turbine Trip failing closed indicates that there is no turbine trip.  Plant document states that this channel is not used for FW control.

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 21 - Turbine Trip (Input) fails open (fails to opposite state)	There is no indication of this failure. However, a reactor trip will occur. In addition, the turbine and reactor trips are annunciated in the PDU.	<p>Plant information states “When the turbine trip signal is active, a digital signal is sent to the digital feedwater microprocessors which process the signal and causes the feedwater regulating valve to ramp shut. At the same time, control of the bypass feedwater regulating valve is changed from the BFV controller (1-FIC-1105, -1106) to the feedwater bypass trip set control (1-FC-1211, -1221). The trip set control provides a constant output signal to the electro-pneumatic converter (1-I/P-1105, -1106) which will position the bypass valve to provide 5 percent of full load feedwater flow.”</p> <p>Accordingly, a reactor trip is expected since the MFRV will ramp shut. The Main CPU will automatically control the DFWCS after the trip.</p>	Input state: open circuit (closed = not tripped). Turbine Trip failing open indicates that there is a turbine trip. Plant information states that this channel is not used for FW control.
Channel 22 - Main CPU Failed (Input) fails closed (fails as is)	A Main CPU failure will be annunciated in the PDU and in the plant computer.	A failover from the Main CPU to the B/U CPU will take place, and the B/U CPU will be in automatic control.	Input state: open circuit (closed = failed). Main CPU Failed (from the MFV) failing closed indicates that the Main CPU failed.
Channel 22 - Main CPU Failed (Input) fails open (fails to opposite state)	There is no indication of this failure.	<p>The Main CPU would “think” that it is OK, regardless of its status. If the Main CPU is OK, operation of DFWCS is unaffected.</p> <p>If the Main CPU is failed, the MFV controller will detect this failure, and the B/U CPU will take control of the DFWCS.</p>	Input state: open circuit (closed = failed). Main CPU Failed (from the MFV) failing open indicates that the Main CPU is OK.



FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 23 - B/U CPU Failed (Input) fails closed (fails as is)	A B/U CPU failure will be annunciated in the PDU and in the plant computer.	The Main CPU believes that the B/U CPU failed. Operation of DFWCS is unaffected. If the Main CPU fails, the MFV controller will detect this failure, but this controller knows that the B/U CPU is OK; hence, the B/U CPU will take control of the DFWCS.	Input state: open circuit (closed = failed). B/U CPU Failed (from the MFV) failing closed indicates that the B/U CPU failed.
Channel 23 - B/U CPU Failed (Input) fails open (fails to opposite state)	There is no indication of this failure.	The Main CPU would “think” that the B/U CPU is OK, regardless of the B/U CPU’s status. If the Main CPU is OK, operation of DFWCS is unaffected.  If the Main CPU failed, a failover to the B/U CPU would occur. If the B/U CPU is OK, this CPU would take automatic control, and operation of DFWCS is unaffected. If the B/U CPU fails, the controllers would go to manual.	Input state: open circuit (closed = failed). B/U CPU Failed (from the MFV) failing open indicates that the B/U CPU is OK.
Channel 24 - Time Sync (Input)	If the time is reset, it may be shown in the PDU.	No effect.	An external clock synchronization signal causes the time to reset to a pre-arranged value defined in the setpoints. Our understanding is that the input “Time Sync” is associated with this signal. It appears that this input is not used in the control of the DFWCS.

FMEA of Digital Backplane (I/O) of Main CPU

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 25 - Neutron Flux # 1 Bypass (Input) fails closed (fails as is)	There is no indication of this failure.	<p>The Main CPU believes the neutron flux # 1 is not bypassed, regardless of the position of the external keyswitch. If the position of the keyswitch is “normal,” i.e., not bypassed, operation of DFWCS is not affected.</p> <p>If the position of the keyswitch is “bypass,” the Main CPU still will use the neutron flux # 1, possibly resulting in incorrect control of the DFWCS.</p>	<p>Input state: open circuit (we assumed closed = not bypassed). Neutron Flux # 1 Bypass failing closed indicates that this flux is not bypassed.</p> <p>An external keyswitch is used to bypass the neutron flux signal.</p>
Channel 25 - Neutron Flux # 1 Bypass (Input) fails open (fails to opposite state)	It appears that the status of the neutron flux signal # 1, i.e., normal or bypass, is not displayed in the PDU. If this assumption is correct, there is no indication of this failure.	<p>The Main CPU believes the neutron flux # 1 is bypassed, regardless of the position of the external keyswitch. The neutron flux signal # 1 will be taken out of service but the other neutron flux signal will be used.</p> <p>If the position of the keyswitch is “bypass,” the Main CPU’s action is appropriate, so operation of the DFWCS is not affected.</p> <p>If the position of the keyswitch is “normal,” i.e., not bypassed, the Main CPU won’t use the neutron flux # 1, resulting in a degradation of the input data used by the DFWCS.</p>	<p>Input state: open circuit (we assumed closed = not bypassed). Neutron Flux # 1 Bypass failing open indicates that this flux is bypassed.</p> <p>An external keyswitch is used to bypass the neutron flux signal.</p>

FMEA of Digital Backplane (I/O) of Main CPU

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 26 - Neutron Flux # 2 Bypass (Input) fails closed (fails as is)	There is no indication of this failure.	<p>The Main CPU believes the neutron flux # 2 is not bypassed, regardless of the position of the external keyswitch. If the position of the keyswitch is “normal,” i.e., not bypassed, operation of DFWCS is not affected.</p> <p>If the position of the keyswitch is “bypass,” the Main CPU still will use the neutron flux # 2, possibly resulting in incorrect control of the DFWCS.</p>	<p>Input state: open circuit (we assumed closed = not bypassed). Neutron Flux # 2 Bypass failing closed indicates that this flux is not bypassed.</p> <p>An external keyswitch is used to bypass the neutron flux signal.</p>
Channel 26 - Neutron Flux # 2 Bypass (Input) fails open (fails to opposite state)	It appears that the status of the neutron flux signal # 2, i.e., normal or bypass, is not displayed in the PDU. If this assumption is correct, there is no indication of this failure.	<p>The Main CPU believes the neutron flux # 2 is bypassed, regardless of the position of the external keyswitch. The neutron flux signal # 2 will be taken out of service but the other neutron flux signal will be used.</p> <p>If the position of the keyswitch is “bypass,” the Main CPU’s action is appropriate, so operation of the DFWCS is not affected.</p> <p>If the position of the keyswitch is “normal,” i.e., not bypassed, the Main CPU won’t use the neutron flux # 2, resulting in a degradation of the input data used by the DFWCS.</p>	<p>Input state: open circuit (we assumed closed = not bypassed). Neutron Flux # 2 Bypass failing open indicates that this flux is bypassed.</p> <p>An external keyswitch is used to bypass the neutron flux signal.</p>

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 27 - Positioner Selected (Input) fails closed (fails as is)	There is no indication of this failure.	<p>The Main CPU will keep the A positioner as the active positioner. If the A positioner is OK, operation of the DFWCS is unaffected.</p> <p>If the accumulated deviation between the MFV demand from the Main CPU and the position of the MFRV exceeds a setpoint value, the opposite positioner (B) will be put into service and the Diagnostic Transfer mode will be shifted to lockout. Operation of the DFWCS is unaffected.</p>	<p>Input state: open circuit (we assumed open = B positioner selected). Positioner Selected failing closed indicates that the A positioner is selected as the active positioner.</p> <p>An analysis of this channel was not found in plant information.</p>
Channel 27 - Positioner Selected (Input) fails open (fails to opposite state)	There is no direct indication of this failure. An indirect indication is that the PDU will show that the active positioner changed from A to B.	<p>The Main CPU will select the B positioner as the active positioner. If the B positioner is OK, operation of the DFWCS is unaffected.</p> <p>If the accumulated deviation between the MFV demand from the Main CPU and the position of the MFRV exceeds a setpoint value, the opposite positioner (A) will be put into service and the Diagnostic Transfer mode will be shifted to lockout. Operation of the DFWCS is unaffected.</p>	<p>Input state: open circuit (we assumed open = B positioner selected). Positioner Selected failing open indicates that the B positioner is selected as the active positioner.</p> <p>An analysis of this channel was not found in plant information.</p>
Channel 28 - No Failures in Other Microprocessor (Input) fails closed (fails as is)	There is no indication of this failure.	<p>The Main CPU believes that the B/U CPU is OK. Operation of the DFWCS is unaffected.</p> <p>If the Main CPU fails, two cases are possible:  1) If the B/U CPU is OK, a failover to the B/U CPU occurs, so operation of DFWCS is unaffected. 2) If the B/U CPU also is failed, the controllers go to manual, so the operators will have to take manual control.</p>	<p>Input state: open circuit (we assumed closed= no failures). No Failures in Other Microprocessor failing closed indicates that there are no failures in the B/U microprocessor.</p> <p>Plant document names this channel "Deviation Alarm Status from Other CPU."</p>

FMEA of Digital Backplane (I/O) of Main CPU			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 28 - No Failures in Other Microprocessor (Input) fails open (fails to opposite state)	There is no indication of this failure.	The Main CPU believes that the B/U CPU is failed. Operation of the DFWCS is unaffected. However, if the Main CPU fails, a failover to the B/U CPU will not occur. The operators will have to take manual control.	Input state: open circuit (we assumed closed= no failures). No Failures in Other Microprocessor failing open indicates that the B/U microprocessor is failed.  Plant document names this channel "Deviation Alarm Status from Other CPU."
Channel 29 - No Deviations in Other Microprocessor (Input) fails closed (fails as is)	There is no indication of this failure.	The Main CPU believes that there are no deviations in the B/U CPU. Operation of the DFWCS is unaffected.  If the Main CPU fails, two cases are possible: 1) If there are no deviations in the B/U CPU, a failover to the B/U CPU occurs, so operation of DFWCS is unaffected. 2) If there are deviations in the B/U CPU, the controllers go to manual, so the operators will have to take manual control.	Input state: open circuit (we assumed closed= no failures). No Deviations in Other Microprocessor failing closed indicates that this status is OK, i.e., there are no deviations in the other microprocessor.  Plant document names this channel "CPU Level Status from Other CPU" and states that it is not used.
Channel 29 - No Deviations in Other Microprocessor (Input) fails open (fails to opposite state)	There is no indication of this failure.	The Main CPU believes that there are deviations in the B/U CPU. Operation of the DFWCS is unaffected. However, if the Main CPU fails, a failover to the B/U CPU will not occur. The operators will have to take manual control.	Input state: open circuit (we assumed closed= no failures). No Deviations in Other Microprocessor failing open indicates that this status is failed, i.e., there are deviations in the other microprocessor.  Plant document names this channel "CPU Level Status from Other CPU" and states that it is not used.

FMEA of Digital Backplane (I/O) of Main CPU

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<p>Channel 30 - Both Level Signals Valid in Other Microprocessor (Input) fails closed (fails as is)</p>	<p>There is no indication of this failure.</p>	<p>The Main CPU believes that both S/G level signals are invalid in the B/U CPU. The status of the level signals in the B/U is used by the Main CPU in its S/G level deviation logic. It appears that the Main CPU would fail itself when it only has one valid level signal and both S/G level signals are invalid in the B/U CPU. If the Main CPU fails due to this reason, there are two cases: 1) if both S/G level signals are valid in the B/U CPU, this CPU takes control, and 2) if both S/G level signals are invalid in the B/U CPU, it appears that the B/U CPU would fail itself, and the operators would have to take manual control.</p>	<p>Input state: open circuit (we assumed closed= invalid). Both Level Signals Valid in Other Microprocessor failing closed indicates that both S/G level signals are invalid in the B/U microprocessor.</p> <p>Plant document names this channel “CPU Steam Flow Status from Other CPU” and states that it is not used.</p>
<p>Channel 30 - Both Level Signals Valid in Other Microprocessor (Input) fails open (fails to opposite state)</p>	<p>There is no indication of this failure.</p>	<p>The Main CPU believes that both S/G level signals are valid in the B/U CPU. The status of the level signals in the B/U is used by the Main CPU in its S/G level deviation logic. If the Main CPU and its own S/G level signals are OK, operation of the DFWCS is unaffected.</p>	<p>Input state: open circuit (we assumed closed= invalid). Both Level Signals Valid in Other Microprocessor failing open indicates that both S/G level signals are valid in the B/U microprocessor.</p> <p>Plant document names this channel “CPU Steam Flow Status from Other CPU” and states that it is not used.</p>

FMEA of Digital Backplane (I/O) of Main CPU

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 31 - Both Steam Flow and Both FW Flow Signals Valid in Other Microprocessor (Input)	NA	NA	Plant document names this channel "CPU Feedflow Status from Other CPU," and states that this channel is not used. This channel is connected to the other CPU, however, it is unknown how the information transmitted through this channel is used by the other CPU.

**Table B.2-4 FMEA of MFV Controller (FIC-1111/1121)**

FMEA of MFV Controller (FIC-1111/1121)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of analog input (Fail to 0.0 VDC)</u>			
ANI0 (S/G level) Fail to 0.0	No alarm or message. The display will be - 116.5".	The display at the MFV controller will be low. The failure can affect the operator's ability to manually control the MFRV.	The signal is for display only.



FMEA of MFV Controller (FIC-1111/1121)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
ANI1 (Valve demand from the main CPU) Fail to 0.0	A deviation alarm will be activated by the MFV controller when the Main CPU demand signal differs from the B/U CPU demand signal by greater than a settable, predetermined setpoint after a settable predetermined time delay. The deviation status will be sent to the BFV controller via Microlink. The BFV controller will activate an alarm to the Plant Computer. The PDI controller will display a "MFV Fail" message.	<p>The controller will initially forward the failed demand signal to the MFRV positioner, PDI controller, and the CPUs of the other S/G. The PDI controller will then detect the signal failure and automatically become the manual controller for the MFV using the old value in its circular buffer. The MFRV must be manually controlled from the PDI controller.</p> <p>The failed signal will be sent to the CPUs of the other S/G, and probably will not affect the FWP speed calculation, because the speed calculation selects the higher of the two flow demand signals, the flow demand signal calculated by the CPUs and the flow demand signal back calculated from the MFV signal received from the other S/G.</p>	<p>The response specified in plant document probably will not take place, because the PDI controller has a scan time of not exceeding 100 milliseconds, while the CPU failover logic has a 1 second delay.</p> <p>The MFV demand signal is also sent to the CPUs of the other S/G and used in the FWP speed calculation of the other S/G.</p>
ANI2 (Valve demand from the B/U CPU) Fail to 0.0	A deviation message is activated, after a settable, predetermined time delay. The deviation message will be sent to the BFV controller through Microlink, and the BFV controller will activate a System Trouble alarm at the Plant Computer.	The MFV controller will continue to forward the signal from the main CPU to its output. No effect on system operation is expected.	

FMEA of MFV Controller (FIC-1111/1121)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of analog output (Fail to 0.0 VDC)</u>			
ANO0 (Output to the MFRV positioner, PDI controller, and other S/G) Fail to 0.0	The PDI controller will display a “MFV Fail” message.	<p>The demand signal to the MFRV positioner will fail to 0, and the valve will begin to shut. The PDI controller will detect the failure and automatically transfer to the MFV Fail mode. The PDI controller output will then rise to the pre-failure value of the MFV controller output and the MFRV will return to that position. The MFRV must be manually controlled from the PDI controller.</p> <p>The failed signal will initially be sent to the CPUs of the other S/G, and probably will not affect the FWP speed calculation.</p>	It is not expected that CPU failover would take place, because the PDI controller would take over.
ANO2 (S/G level setpoint output) Fail to 0.0	A system deviation alarm at the Plant Computer will be activated, if a setpoint deviation is detected. The setpoint display at the BFV controller will be low.	The CPUs may detect a setpoint deviation if the deviation setpoint limit is exceeded, and revert to a built-in setpoint.	The operator may use the MFV controller to manually adjust the SG level setpoint.

FMEA of MFV Controller (FIC-1111/1121)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Digital Inputs Fail Open</u>			
CCI0 (B/U CPU Power Fail or in Test) Fails Open	The controller will indicate that the B/U CPU is failed, and the B/U CPU status will be sent through Microlink to the BFV controller which will activate an annunciator in the control room.	The controller will block the B/U CPU demand signal from its output. System operation will not be affected. The B/U CPU status is sent to the CPUs and could affect the deviation logic of the CPUs.	The signal is normally closed indicating the B/U CPU is OK.  It is not clear what the B/U CPU would do when it receives the failure status of its own from the MFV controller. How does the B/U CPU determine its status to send to the Main CPU?
CCI1 (B/U CPU Fail) Fails Open	None.	The controller will not be able to determine the correct status of the B/U CPU. The operation is not affected unless other failures occur.	The signal is normally open indicating the B/U CPU is OK.
CCI2 (Main CPU Power Fail or in Test) Fails Open	The BFV controller will actuate an alarm to the Plant Computer.	Failover from the main CPU to the B/U CPU will take place. The controller will send a Main CPU Fail signal to the CPUs and to the BFV controller through Microlink. The Main CPU Fail signal affects deviation logic of the B/U CPU.	The signal is normally closed indicating the Main CPU is OK.  It is not clear what the Main CPU will do when it receives the Main CPU Fail signal from the MFV controller. How does the Main CPU determine its own status to send to the B/U CPU?
CCI3 (Main CPU Fail) Fails Open	None.	The controller will not be able to determine the status of the Main CPU. The operation is not affected unless other failures occur.	The signal is normally open indicating the main CPU is OK.
<u>Digital Input Fail Closed</u>			

FMEA of MFV Controller (FIC-1111/1121)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCI0 (B/U CPU Power Fail or in Test) Fails Closed	None.	The controller will not be able to determine the correct status of the B/U CPU. The operation is not affected unless other failures occur.	The signal is normally closed indicating the B/U CPU is OK.
CCI1 (B/U CPU Fail) Fails Closed	The controller will indicate that the B/U CPU is failed, and the B/U CPU status will be sent through Microlink to the BFV controller which will activate an annunciator in the control room.	The controller will block the B/U CPU demand signal from its output. System operation will not be affected. The B/U CPU status is sent to the CPUs and could affect the deviation logic of the CPUs.	It is not clear what the B/U CPU would do when it receives the failure status of its own from the MFV controller. How does the B/U CPU determine its status to send to the Main CPU?  The signal is normally open indicating that the CPU is OK.
CCI2 (Main CPU Power Fail or in Test) Fails Closed	None.	The controller will not be able to determine the correct status of the Main CPU. The operation is not affected unless other failures occur.	The signal is normally closed.
CCI3 (Main CPU Fail) Fails Closed	The BFV controller will actuate an annunciator in the control room indicating the Main CPU Fail.	A failover from the Main CPU to the B/U CPU will take place. The controller will send a Main CPU Fail signal to the CPUs and to the BFV controller through Microlink. The Main CPU Fail signal affects deviation logic of the B/U CPU.	The signal is normally open indicating the main CPU is OK.  It is not clear what the B/U CPU would do when it receives the failure status of its own from the MFV controller. How does the Main CPU determine its own status to send to the B/U CPU?

FMEA of MFV Controller (FIC-1111/1121)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Digital Outputs Fail Open</u>			
CCO0 (A/M Status to the Main CPU) Fails Open	The PDU of the Main CPU will display the Transfer Inhibit Alarm. The alarm will also be sent to the Plant Computer.	<p>A manual signal will be sent to the Main CPU, and the Transfer Inhibit Alarm window will be activated. Assuming the Main CPU is in control, and the MFV controller is in auto, the Main CPU will track the MFV controller output. The MFV controller output will be sent from the Main CPU to the MFV controller. The automatic control is effectively lost. This failure may lead to a reactor trip.</p> <p>Normally, upon a reactor trip, the MFRV will be ramped closed and the post trip positioning relay circuit will ensure the MFV demand signal is reduced to zero. It is not obvious that the MFRV will be ramped closed, when the controller is in Manual. The post trip positioning relay circuit should ensure the MFRV be closed.</p>	<p>The signal is normally closed when in auto mode.</p> <p>The response to a reactor trip needs to be confirmed through review of the software.</p>
CCO1 (A/M Status to the B/U CPU) Fails Open	The PDU of the B/U CPU will display the Transfer Inhibit Alarm. The alarm will also be sent to the Plant Computer.	Assuming the Main CPU is in control and the controller is in auto, the operation will not be affected.	The signal is normally closed when in auto mode.

FMEA of MFV Controller (FIC-1111/1121)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCO2 (B/U CPU Failed Status to CPUs) Fails Open	There is no direct indication of the failure.	<p>The failed signal will be sent to the Main and B/U CPUs.</p> <p>Assuming the Main CPU is in control, and the controller is in auto, the operation is not affected.</p>	The signal is normally open indicating the B/U CPU is OK.
	If the MFV controller detects failure of the B/U CPU, it generates a local B/U CPU Fail message and sends the status through Microlink to the BFV controller which will actuate an annunciator in the control room.	Assuming the Main CPU is not available, and the B/U CPU is in control, when the failure occurs, the MFV controller should know the correct status of the B/U CPU, and use the MFV demand from the B/U CPU as the output. The system operation will not be affected. If, in addition, the B/U CPU fails, the MFV controller should be able to detect it and transfer to the manual mode.	We assumed that the failure mode is a local failure of the output circuitry, not the controller itself.

FMEA of MFV Controller (FIC-1111/1121)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCO3 (Main CPU Failed Status to CPUs) Fails Open	There is no direct indication of the failure.	<p>The failed signal will be sent to the Main and B/U CPUs.</p> <p>Assuming the Main CPU is in control, the operation is not affected.</p>	This signal is normally open indicating the Main CPU is OK.
	If the MFV controller detects failure of the Main CPU, it generates a local Main CPU Fail message and sends the status through Microlink to the BFV controller which will actuate an annunciator in the control room.	Assuming the Main CPU failed while in control, its failure should be detected by the MFV controller, and a failover to the B/U CPU will take place. The incorrect Main CPU status may affect the deviation logic of the B/U CPU.	<p>We assumed that the failure mode is a local failure of the output circuitry, not the controller itself.</p> <p>It is not clear how the B/U CPU reconciles the conflicting information about status of the Main CPU; that is, the MFV controller indicates it is good, while the signal directly from the Main CPU probably indicates it has failed.</p>

FMEA of MFV Controller (FIC-1111/1121)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Digital Outputs Fail Closed</u>			
CCO0 (A/M Status to the Main CPU) Fails Closed	No direct indication of the failure is available.	The failed signal will be sent to the Main CPU. If the Main CPU is in control, the system operation is not affected. If the operator switches the controller to manual, the Main CPU will not recognize it, and continues sending its output to the MFV. As a result, Transfer Inhibit will not be activated. As long as the operator properly takes control, the operation continues until the MFV output deviation from the Main CPU output exceeds the setpoint, in which case a failover from Main CPU to B/U CPU takes place. If the operator fails to manually control MFV, a loss of feedwater control may lead to a reactor trip. Is it possible that a Transfer is initiated with the failure? Upon a reactor trip, the MFRV will be ramped closed and the post trip positioning relay circuit will ensure the MFV demand signal is reduced to zero. The pre-existing failure of the CCO0 does not affect the response to a reactor trip.	The signal is normally closed when in auto mode.



FMEA of MFV Controller (FIC-1111/1121)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCO1 (A/M Status to the B/U CPU) Fails Closed	No direct indication of the failure is available.	If the Main CPU is in control, and the controller is in auto, then the system operation is not affected.	The signal is normally closed when the controller is in auto.
	The deviation will actuate an alarm be sent to the Plant Computer.	If the B/U CPU is in control, and the operator changes the controller to manual, the B/U CPU will not be able to detect it, and the Transfer Inhibit will not be actuated. The B/U CPU continues sending its MFV demand to the controller until the deviation between the MFV demand calculated by the B/U CPU and the MFV controller output exceeds the setpoint, when the B/U CPU will fail and the MFV controller will transfer to manual.	
CCO2 (B/U CPU Failed Status to CPUs) Fails Closed	No direct indication of the failure will be available.	The failed signal will be sent to both CPUs. The MFV controller itself is aware of the correct status of the B/U CPU.  If the Main CPU is in control and the controller is in Auto, system operation will not be affected. The failed signal may affect the deviation logic of the Main CPU.	The signal is normally open indicating the B/U CPU is OK.  It is not clear how the Main CPU reconciles the conflicting information about status of the B/U CPU; that is, the MFV controller indicates it is failed, while the signal directly from the B/U CPU probably indicates it is good.
		If the B/U CPU is not failed, the Main CPU is failed, and the controller is in Auto, the failure will cause the controllers to switch to manual.	It is assumed that the B/U CPU will fail itself.

FMEA of MFV Controller (FIC-1111/1121)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCO3 (Main CPU Failed Status to CPUs) Fails Closed	The Main CPU failure will be annunciated in the control room.	<p>The failed signal will be sent to both CPUs. The MFV controller itself is aware of the correct status of the B/U CPU.</p> <p>If the Main CPU is in control, and the controller is in Auto, a failover to B/U CPU will take place.</p>	<p>The signal is normally open indicating the Main CPU is OK.</p> <p>It is assumed that the Main CPU will fail itself when it receives the failed signal.</p>
<u>Loss of Power to Controller</u>			
Loss of power	The MFV controller will be off. The PDI controller will display a "MFV Fail" message.	<p>All analog outputs fail to 0.</p> <p>All digital outputs fail to Open status.</p> <p>The PDI controller will automatically switch to its MFV failure mode of operation and its output will raise to the pre-failure output level of the MFV controller. The MFRV has to be controlled manually using the PDI controller.</p> <p>The CPUs will use the built-in S/G level setpoint and track PDI controller output.</p>	

**Table B.2-5 FMEA of BFV Controller (FIC-1105/1106)**

In conducting the FMEA of the BFV Controller we made the following assumptions:

1. We assumed that initially the DFWCS is in automatic high-power mode with all system modules normally running, and the Main CPU is controlling the feedwater system. It appears that plant hazards analysis of the BFV controller assuming that the DFWCS is operating in low-power mode.
2. The BFRV is normally closed during high-power mode. A signal of "0.0" to the BFRV positioner is interpreted in this analysis to result in the BFRV remaining closed.

FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of Analog Input (Fail to 0.0 VDC)</u>			
ANI0 (Steam generator (S/G) level) fails to 0.0 VDC	The controller will display a value and bargraph of S/G level equal to -116.5". No alarms will be activated.	The DFWCS will continue its operation in automatic mode.	The S/G level signal is used for display only.

FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
ANI1 (Valve demand from the main CPU) fails to 0.0 VDC	During normal high-power mode (i.e., not high-power override mode) the failure is not detected.	<p>The controller will forward the failed demand signal to the BFRV positioner per automatic mode of control. Since the signal corresponds to closure of the BFRV, and the BFRV is already closed, there is no negative effect on the operation of the DFWCS.</p> <p>The failure would manifest when the BFRV should open, but would receive a signal to remain closed. The BFRV is required to open when there is a transfer to 1) low-power mode, or 2) high-power override mode. A deviation would be detected by the Main CPU which will then fail. Subsequently, the backup (B/U) CPU also will fail due to the same reason. Hence, there would be a loss of automatic control of the DFWCS.</p>	<p>The BFRV is normally closed during high-power mode.</p> <p>Input signals to BFV controller are clamped within their range limits. This appears to mean that a failure to 0.0 VDC of ANI1 will be interpreted by the controller as a signal to close the BFRV.</p> <p>The PDI controller also receives the failed demand signal which is held in a circular buffer.</p>
ANI2 (S/G level setpoint) fails to 0.0 VDC	The controller will display a value and bargraph of S/G level equal to -116.5". No alarms will be activated.	The DFWCS will continue its operation in automatic mode.	The level setpoint signal is used for display only.

FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
ANI3 (Valve demand from the B/U CPU) fails to 0.0 VDC	During high-power mode with no additional failures, the failure is not detected (see comments).	<p>Since this signal is not used when the main CPU is controlling, there is no negative effect on the operation of the DFWCS.</p> <p>The failure would manifest when 1) the main CPU fails, and 2) the BFRV is required to open, but would receive a signal to remain closed. The BFRV is required to open when there is a transfer to 1) low-power mode, or 2) high-power override mode. A deviation would be detected by the Main CPU which will then fail. Subsequently, the backup CPU also will fail due to the same reason. Hence, there would be a loss of automatic control of the DFWCS.</p>	<p>Normally, the BFV controller sends the demand from the main CPU to the BFRV positioner.</p> <p>Plant information indicates that the BFV controller will detect the deviation between the main and backup CPU demand signals when they differ by greater than a settable, predetermined setpoint after a settable, predetermined time delay. However, since both signals demand the BFRV to be closed, it is not clear that the failure will be detected.</p>

FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of Analog Output (Fail to 0.0 mADC)</u>			
ANO0 (Output to the BFRV (positioner)) fails to 0.0 mADC	During normal high-power mode (i.e., not high-power override mode) the failure is not detected.	<p>Since the signal corresponds to closure of the BFRV, and the BFRV is already closed, there is no negative effect on the operation of the DFWCS.</p> <p>The failure would manifest when the BFRV should open, but would receive a signal to remain closed. The BFRV is required to open when there is a transfer to 1) low-power mode, or 2) high-power override mode. A deviation would be detected by the Main CPU which will then fail. Subsequently, the backup CPU also will fail due to the same reason. Hence, there would be a loss of automatic control of the DFWCS.</p>	<p>The BFRV is normally closed during high-power mode.</p> <p>The PDI controller also receives the failed demand signal which is held in a circular buffer.</p>

FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Digital Input (Fail Open)</u>			
CCI0 (B/U CPU Power Fail or in Test) fails open	The controller will display the message “B/U FAIL.” The “Main or B/U CPU Fail” contact (CCO3) shall close, so a plant computer DFWCS Trouble Alarm will actuate.	<p>The controller will block the B/U CPU BFRV demand signal from its output. As long as the Main CPU is available, system operation will be unaffected and the Main CPU BFRV demand signal will continue to be forwarded to the output.</p> <p>If the Main CPU is not available, the BFV controller will indicate that both CPUs are failed and will revert to Manual mode of operation. The operator will then be required to take action to control S/G level.</p>	Contact CCI0 open means that the Power Fail / Test status of the B/U CPU failed.
CCI1 (B/U CPU Fail) fails open	This failure cannot be detected. If the B/U CPU actually fails its watchdog test, the failure will be detected by other controllers that, in turn, will send a “B/U CPU Fail” signal to the BFV controller. In this way, a plant computer DFWCS Trouble Alarm will actuate.	<p>The controller will be unable to determine the watchdog status of the B/U CPU. The controller will assume that the watchdog status is normal.</p> <p>System operation is unaffected unless the B/U CPU actually fails its watchdog test (which will be detected) and the Main CPU becomes unavailable. When this happens, the BFRV demand signal from the failed B/U CPU will be sent to the BFRV positioner. The impact on the DFWCS will vary depending on the nature and severity of the B/U CPU fault.</p>	Contact CCI1 open means that the watchdog status of the B/U CPU is OK.

FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCI2 (Main CPU Power Fail or in Test) fails open	The controller will display the message "M FAIL." The "Main or B/U CPU Fail" contact (CCO3) shall close, so a plant computer DFWCS Trouble Alarm will actuate.	The controller will block the Main CPU BFRV demand signal and will forward the "tracking" B/U CPU BFRV demand signal to its output. The Main and B/U CPUs will remain in controlling and tracking modes, respectively. The controller's output will drift upward or downward. This may result in the BFRV opening to some extent.	Contact CCI2 open means that the Power Fail / Test status of the Main CPU failed.  Plant information indicates that if the output drifts beyond the deviation limit of the Main CPU, it will fail, and the B/U CPU will assume automatic control of the BFRV. However, it appears that the Main CPU will not fail because the CPU deviation logic for the BFRV demand signal is inhibited during High Power Mode Operations.
CCI3 (Main CPU Fail) fails open	This failure cannot be detected. If the Main CPU actually fails its watchdog test, the failure will be detected by other controllers that, in turn, will send a "Main CPU Fail" signal to the BFV controller. In this way, a plant computer DFWCS Trouble Alarm will actuate.	The controller will be unable to determine the watchdog status of the Main CPU. The controller will assume that the watchdog status is normal.  System operation is unaffected unless the Main CPU actually fails its watchdog test (which will be detected). When this failure occurs, the BFRV demand signal from the failed Main CPU will be sent to the BFRV positioner. The impact on the DFWCS will vary depending on the nature and severity of the Main CPU fault.	Contact CCI3 open means that the watchdog status of the Main CPU is OK.



FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Digital Input (Fail Closed)</u>			
CCI0 (B/U CPU Power Fail or in Test) fails closed	This failure cannot be detected. If the B/U CPU actually fails or is placed in test, the failure will be detected by other controllers that, in turn, will send a "B/U CPU Fail" signal to the BFV controller. In this way, a plant computer DFWCS Trouble Alarm will actuate.	<p>The controller will be unable to determine the Power Fail / Test status of the B/U CPU. The controller will assume that this status is normal.</p> <p>System operation is unaffected unless 1) the B/U CPU actually fails or is placed in test (either of these events will be detected by the MFV controller which will send a failure signal to B/U CPU), and 2) the Main CPU becomes unavailable. When this happens, the BFRV demand signal from the failed B/U CPU will be sent to the BFRV positioner. The impact on the DFWCS will vary depending on the nature and severity of the B/U CPU fault.</p>	Contact CCI0 closed means that the Power Fail / Test status of the B/U CPU is OK.
CCI1 (B/U CPU Fail) fails closed	The controller will display the message "B/U FAIL." The "Main or B/U CPU Fail" contact (CCO3) shall close, so a plant computer DFWCS Trouble Alarm will actuate.	<p>The controller will block the B/U CPU BFRV demand signal from its output.</p> <p>As long as the Main CPU is available, system operation will be unaffected and the Main CPU BFRV demand signal will continue to be forwarded to the output.</p> <p>If the Main CPU is not available, the BFV controller will indicate that both CPUs are failed and will revert to Manual mode of operation. The operator will then be required to take action to control S/G level.</p>	Contact CCI1 closed means that the watchdog status of the B/U CPU is failed.

FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCI2 (Main CPU Power Fail or in Test) fails closed	This failure cannot be detected. If the Main CPU actually fails or is placed in test, the failure will be detected by other controllers that, in turn, will send a "Main CPU Fail" signal to the BFV controller. In this way, a plant computer DFWCS Trouble Alarm will actuate.	<p>The controller will be unable to determine the Power Fail / Test status of the Main CPU. The controller will assume that this status is normal.</p> <p>System operation is unaffected unless the Main CPU actually fails or is placed in test. Either of these events will be detected by the MFV controller which will send a failure signal to the Main CPU. When either of these events occurs, the BFRV demand signal from the failed Main CPU will be sent to the BFRV positioner. The impact on the DFWCS will vary depending on the nature and severity of the Main CPU fault.</p>	Contact CCI2 closed means that the Power Fail / Test status of the Main CPU is OK.
CCI3 (Main CPU Fail) fails closed	The controller will display the message "M FAIL." The "Main or B/U CPU Fail" contact (CCO3) shall close, so a plant computer DFWCS Trouble Alarm will actuate.	The controller will block the Main CPU BFRV demand signal and will forward the "tracking" B/U CPU BFRV demand signal to its output. The Main and B/U CPUs will remain in controlling and tracking modes, respectively. The controller's output will drift upward or downward. This may result in the BFRV opening to some extent.	<p>Contact CCI3 closed means that the watchdog status of the Main CPU is failed.</p> <p>Plant information indicates that if the output drifts beyond the deviation limit of the Main CPU, it will fail, and the B/U CPU will assume automatic control of the BFRV. However, it appears that the Main CPU will not fail because the CPU deviation logic for the BFRV demand signal is inhibited during High Power Mode Operations.</p>

FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Digital Output (Fail Open)</u>			
CCO0 (Auto/Manual Status to the Main CPU) fails open	<p>Plant analysis states that "...the Transfer Inhibit Alarm window will be activated." It appears that this "window" refers to an annunciator in the main control room.</p> <p>The CPUs include a digital output to provide indication for the plant computer whenever automatic valve transfer is inhibited.</p>	<p>A Manual status signal will be sent to the DFWCS Main CPU regardless of the actual status of the controller. Thus, the transfer of high power to low power mode is inhibited. If the controller is in Manual mode, or the B/U CPU is controlling S/G level, operation is unaffected.</p> <p>If the controller is in Auto mode, and the main CPU is controlling S/G level, this CPU will "think" that the controller is in Manual mode, so it appears that it (and the B/U CPU) will track the BFRV demand from the controller's output. The controller, in turn, will receive the tracked signal, and forward it to its output. The controller's output will drift upward or downward. This may result in the BFRV opening to some extent.</p>	<p>Contact CCO0 open means that the Auto/Manual Status to the Main CPU indicates Manual.</p> <p>Plant information states that "Main CPU Automatic control of S/G level is lost during this failure." However, the Main CPU will keep Automatic control of the rest of the modules of the DFWCS, so it appears that this CPU can remain in control of S/G level, unless there are additional failures.</p>

FMEA of BFV Controller (FIC-1105/1106)

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<p>CCO1 (Auto/Manual Status to the B/U CPU) fails open</p>	<p>Plant analysis states that "...the Transfer Inhibit Alarm window will be activated." It appears that this "window" refers to an annunciator in the main control room.</p> <p>The CPUs include a digital output to provide indication for the plant computer whenever automatic valve transfer is inhibited.</p>	<p>A Manual status signal will be sent to the DFWCS B/U CPU regardless of the actual status of the controller. Thus, the transfer of high power to low power mode is inhibited. If the controller is in Manual mode, or the Main CPU is controlling S/G level, operation is unaffected.</p> <p>If the controller is in Auto mode, and the B/U CPU is controlling S/G level, this CPU will "think" that the controller is in Manual mode, so it appears that it (and the Main CPU if available) will track the BFRV demand from the controller's output. The controller, in turn, will receive the tracked signal and forward it to its output. The controller's output will drift upward or downward. This may result in the BFRV opening to some extent.</p> <p>B/U CPU Automatic control of S/G level is lost during this failure if operating in low power mode.</p>	<p>Contact CCO1 open means that the Auto/Manual Status to the B/U CPU indicates Manual.</p>

FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCO2 (Main and B/U CPUs Failed Status) fails open	A status signal of "Both CPUs OK" will be sent to the Fail to Manual Alarm window (annunciator), regardless of the actual status of both CPUs. There is no detection of the contact CCO2 failing open.	<p>If either the Main or the B/U CPU (or both) is OK, then the signal is correct. The operation of the DFWCS is unaffected.</p> <p>If both CPUs failed, the Fail to Manual Alarm annunciator is incorrect. This annunciation ("Both CPUs OK") would fail to alert the operators to take manual control of the DFWCS. The DFWCS would not be controlled neither automatically nor manually. It is not known at this time the consequences of this total loss of control.</p>	Contact CCO2 open means that the Main and B/U CPUs Failed Status is OK, i.e., at least one CPU is not failed.
CCO3 (Main or B/U CPU Failed Status) fails open	A status signal of "CPU OK" will be sent to the Plant Computer, regardless of the actual status of each CPU. There is no detection of the contact CCO3 failing open.	<p>If both CPUs are OK, the signal is correct. The operation of the DFWCS is unaffected.</p> <p>If the main (B/U) CPU is failed, the signal is incorrect. However, the DFWCS is controlled by the B/U (main) CPU.</p> <p>If both CPUs failed, the Fail to Manual Alarm annunciator (fed from CCO2) would alert the operators to take manual control of the DFWCS.</p>	Contact CCO3 open means that the Main or B/U Failed Status is OK, i.e., both CPUs are OK (not failed).

FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Digital Output (Fail Closed)</u>			
CCO0 (Auto/Manual Status to the Main CPU) fails closed	This failure mode is not detected.	<p>An Auto status signal will be sent to the DFWCS Main CPU regardless of the actual status of the controller. Operation is unaffected if the controller is in Auto mode, or the B/U CPU is in control.</p> <p>If the controller is in Manual mode, and the Main CPU “thinks” it is in control (due to the erroneous signal), this CPU will attempt to control the BFRV by keeping it closed, even though the BFV controller blocks this CPU’s signal when it’s in manual mode. Operation of DFWCS is unaffected.</p>	<p>Contact CCO0 closed means that the Auto/Manual Status to the Main CPU indicates Automatic.</p> <p>Plant information indicates that the Main CPU will fail when the Deviation Setpoint is reached. However, it appears that the Main CPU will not fail because the CPU deviation logic for the BFRV demand signal is inhibited during High Power Mode Operations.</p>
CCO1 (Auto/Manual Status to the B/U CPU) fails closed	This failure mode is not detected.	<p>An Auto status signal will be sent to the DFWCS B/U CPU regardless of the actual status of the controller. If the controller is in Auto mode, or the Main CPU is in control, operation is unaffected.</p> <p>If the controller is in Manual mode, and the Main CPU failed, the B/U CPU “thinks” that it is in control, so this CPU will attempt to control the BFRV by keeping it closed, but the BFV controller blocks this CPU’s signal when it’s in manual mode. Operation of DFWCS is unaffected.</p>	<p>Contact CCO1 closed means that the Auto/Manual Status to the B/U CPU indicates Automatic.</p> <p>Plant information indicates that the B/U CPU will fail when the Deviation Setpoint is reached. However, it appears that this CPU will not fail because the CPU deviation logic for the BFRV demand signal is inhibited during High Power Mode Operations.</p>

FMEA of BFV Controller (FIC-1105/1106)

Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCO2 (Main and B/U CPUs Failed Status) fails closed	A status signal of “Both CPUs Failed” will be sent to the Fail to Manual Alarm window (annunciator), regardless of the actual status of both CPUs. This annunciator will actuate.	<p>If both the Main and B/U CPUs are OK, the DFWCS is controlled in the Automatic mode. The incorrect signal may be puzzling to the operators. However, the “Main or B/U CPU Failed Status” (from CCO3) indicates that no CPU failed; this indication, in turn, would give a clue to the operators that the incorrect signal is wrong. Nevertheless, the operators may decide to take manual control of the DFWCS. In this way, errors may be executed. If either the Main or the B/U CPU (but not both) failed, the DFWCS is controlled in the Automatic mode by the remaining CPU. Both the “Main and B/U CPUs Failed Status” and “Main or B/U CPU Failed Status” indicate failure. The operators are likely to take manual control of the DFWCS. In this way, errors may be executed.</p> <p>If both CPUs failed, the signal is correct. Operation of the DFWCS is unaffected, except for the failure of the CPUs.</p>	Contact CCO2 closed means that the Main and B/U CPUs Failed Status is failed, i.e., the Main and B/U CPUs are failed.

FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCO3 (Main or B/U CPU Failed Status) fails closed	A status signal of "CPU Failed" will be sent to the Plant Computer, regardless of the actual status of each CPU. The Plant Computer DFWCS Trouble Alarm will actuate.	<p>If both CPUs are OK, the signal is incorrect. However, the operation of the DFWCS is unaffected. The operators are expected to become aware of the Plant Computer DFWCS Trouble Alarm, and troubleshoot this erroneous signal.</p> <p>If the main or B/U CPU is failed, the signal is correct. The operation of the DFWCS is unaffected, except for the failure of one CPU.</p>	Contact CCO3 closed means that the Main or B/U CPU Failed Status is failed, i.e., at least one CPU failed.



FMEA of BFV Controller (FIC-1105/1106)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of Power to Controller</u>			
Loss of power to controller	The display of the controller will be off.	<p>ANO0 fail to 0.0 mADC: Since the signal corresponds to closure of the BFRV, and the BFRV is already closed, there is no negative effect on the operation of the DFWCS. The failure would cause a negative impact when the BFRV should open, but would receive a signal to remain closed. The BFRV is required to open when there is a transfer to other power modes, such as low-power mode.</p> <p>CCO0 (CCO1) open: A Manual status signal will be sent to the DFWCS Main (B/U) CPU regardless of the actual status of the controller.</p> <p>CCO2 open: A status signal of "Both CPUs OK" will be sent to the Fail to Manual Alarm annunciator, regardless of the actual status of both CPUs.</p> <p>CCO3 open: A status signal of "CPU OK" will be sent to the Plant Computer, regardless of the actual status of each CPU.</p> <p>Summary: please continue after * in the column "Comments".</p>	<p>The controller's analog output ANO0 will fail to 0.0 mADC, and the controller's digital outputs will fail to Open status.</p> <p>* Summary: Main and B/U CPUs will receive signals that controller is in manual. Thus, the automatic transfer of power modes is inhibited. The BFRV remains closed due to closure signal. The operators cannot take manual control of the BFRV using its controller. To control the BFRV using the PDI controller, the operators have to position the handswitch HS-4516(17)C in the "Bypass Fail" position. The DFWCS is unable to annunciate failures via BFV controller's contacts CCO2 and CCO3.</p>

**Table B.2-6 FMEA of FWP Controller (FIC-4516/4517)**

FMEA of FWP Controller (FIC-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of analog input (Fail to 0.0 VDC)</u>			
ANI0 (Main CPU Speed Demand) Fails to 0.0	<p>The display at the FWP controller will be low.</p> <p>A deviation alarm is activated at the controller when the Main CPU demand signal differs from the B/U CPU demand signal by greater than a set-point, after a time delay. The deviation alarm status will be sent to the BFV controller which will send the alarm to the Plant Computer (PC).</p> <p>The CPU failures and deviation will be annunciated in the control room and sent to the PC.</p>	<p>The failed signal will be sent to the Lovejoy FWP speed controller which will detect the failure and maintain the FWP speed at pre-failure value.</p> <p>The failed signal is sent to the CPUs for tracking, and after a delay will cause the CPUs to be failed due to deviation logic. As a result, the MFV, BFV and FWP controllers will transfer to manual control. It is not likely that the FWP controller can be used to manually control the FWP in this condition.</p>	Need to confirm operation of the Lovejoy controller.

FMEA of FWP Controller (FIC-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
ANI2 (Bias Signal from Potentialmeter, also sent to the CPUs) Fails to 0.0	The BFV controller will send an alarm to the Plant Computer, upon receipt of the Bias Potential Rate Alarm from the FWP controller.	The failed signal corresponds to a -100% bias. The rate of change of the bias is monitored by the FWP controller, and if a pre-set limit is exceeded, the FWP controller switches to manual mode with the pre-failure value, and a Bias Potential Rate Alarm signal is sent to the BFV controller via the Microlink connection. The BFV controller will then send the alarm to the Plant Computer.	The bias signal is also sent to the Main and B/U CPUs where it is added to the calculated pump speed. It is assumed that the failure is a local failure and a correct signal is sent to the CPUs.
ANI3 (B/U CPU Speed Demand) Fails to 0.0	A deviation alarm at the controller is activated when the main CPU demand signal differs from the B/U CPU demand signal by greater than a settable, predetermined setpoint after a time delay. The deviation alarm is also sent to the BFV controller via Microlink, and the BFV controller will send it to the Plant Computer.	The controller will continue sending the demand from the Main CPU to its output, and the system operation is not affected.	

FMEA of FWP Controller (FIC-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of analog output (Fail to 0.0 VDC)</u>			
ANO0 (Output to the Lovejoy Control System) Fails to 0.0	The CPU failures and deviation will be detected by the BFV controller which will activate an annunciator in the control room and send the alarm to the Plant Computer.	<p>The failed signal will be sent to the Lovejoy FWP speed controller which will detect the failure and maintain the FWP speed at pre-failure value.</p> <p>The failed signal is sent to the CPUs for tracking, and after a time delay will cause the CPUs to be failed due to deviation logic. As a result, the MFV, BFV and FWP controllers will transfer to manual control. A complete loss of automatic control will take place. It is not likely that the FWP controller can be used to manually control the FWP in this condition. The FWP has to be manually controlled using the Lovejoy controller.</p>	Need to confirm operation of the Lovejoy controller.
ANO2 (Bias Potential Excitation) Fails to 0.0  (This failure mode is also applicable to failure to 0.0 of the potential meter.)	The BFV controller will send an alarm to the Plant Computer, upon receipt of the Bias Potential Rate Alarm from the FWP controller.	The failed signal corresponds to a -100% bias. The rate of change of the bias is monitored by the FWP controller, and if a pre-set limit is exceeded, the FWP controller switches to manual mode with the pre-failure value, and a Bias Potential Rate Alarm signal is sent to the BFV controller via the Microlink connection. The BFV controller will then send the alarm to the Plant Computer.	The failed bias signal is also sent to the Main and B/U CPUs where it is added to the calculated pump speed. At the CPU, a FWP bias deviation logic is used to detect out of range condition of the signal. It is probably not going to initiate an alarm, because the bias should be in the expected range. The output of the CPUs will not be used by the FWP controller which is in manual.
<u>Digital Inputs Fail Open</u>			

FMEA of FWP Controller (FIC-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCI0 (B/U CPU Power Fail or in Test) Fails Open	The controller will indicate that the B/U CPU is failed, and the B/U CPU status will be sent through Microlink to the BFV controller which will activate an alarm to the Plant Computer.	The controller will block the B/U CPU demand signal from its output. System operation will not be affected.	The signal is normally closed indicating the B/U CPU is OK.  The B/U CPU status is not sent back to the CPUs. This is true for the BFV controller also.
CCI1 (B/U CPU Fail) Fails Open	None.	The operation is not affected unless other failures occur.	The signal is normally open indicating the B/U CPU is OK.
CCI2 (Main CPU Power Fail or in Test) Fails Open	The BFV controller will actuate an alarm to the Plant Computer.	Failover from the main CPU to the B/U CPU will take place. The controller will send a Main CPU Fail signal to the BFV controller through Microlink. The Main CPU status is not sent back to the CPUs and the CPUs do not know that the controller thinks the Main CPU has failed. The Main CPU continues thinking it is in control, and the B/U CPU continues tracking the output of the controller. Therefore, the FWP demand may remain unchanged, i.e., a loss of automatic control, until the Main CPU detects a deviation and fails itself, and the B/U CPU takes over. It is probably not likely that a reactor trip takes place due to loss of FWP control.	The signal is normally closed indicating the Main CPU is OK.  It is assumed that the Main CPU status information to other controllers is correct.
CCI3 (Main CPU Fail) Fails Open	None.	The controller does not have the correct status of the Main CPU. The operation is not affected unless other failures occur.	The signal is normally open indicating the main CPU is OK.

FMEA of FWP Controller (FIC-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Digital Input Fail Closed</u>			
CCI0 (B/U CPU Power Fail or in Test) Fails Closed	None.	The controller does not have the correct status of the B/U CPU. The operation is not affected unless other failures occur.	The signal is normally closed indicating the B/U CPU is OK.
CCI1 (B/U CPU Fail) Fails Closed	The controller will indicate that the B/U CPU is failed, and the B/U CPU status will be sent through Microlink to the BFV controller which will activate an alarm to the Plant Computer.	The controller will block the B/U CPU demand signal from its output. System operation will not be affected unless other failures take place.	The signal is normally open indicating that the CPU is OK.
CCI2 (Main CPU Power Fail or in Test) Fails Closed	None.	The controller does not have the correct status of the Main CPU. The operation is not affected unless other failures occur.	The signal is normally closed.

FMEA of FWP Controller (FIC-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCI3 (Main CPU Fail) Fails Closed	The BFV controller will actuate an annunciator in the control room indicating the Main CPU Fail.	Failover from the main CPU to the B/U CPU will take place. The controller will send a Main CPU Fail signal to the BFV controller through Microlink. The Main CPU status is not sent back to the CPUs and the CPUs do not know that the controller thinks the Main CPU has failed. The Main CPU continues thinking it is in control, and the B/U CPU continues tracking the output of the controller. Therefore, the FWP demand may remain unchanged, i.e., a loss of automatic control, until the Main CPU detects a deviation and fails itself, and the B/U CPU takes over. It is probably not likely that a reactor trip takes place due to loss of FWP control.	The signal is normally open indicating the main CPU is OK.  It is assumed that the Main CPU status information to other controllers is correct.

FMEA of FWP Controller (FIC-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Digital Outputs Fail Open</u>			
CCO0 (A/M Status to the Main CPU) Fails Open	None.	A Manual status signal will be sent to the Main CPU. Assuming the Main CPU is in control, and the FWP controller is in auto, the Main CPU will switch to tracking mode and continue sending its output to the FWP controller, with the controller remaining in Auto. The B/U CPU will continue its tracking also. There will be no Transfer Inhibit Alarm. The automatic control is effectively lost. The output of the controller may drift with no direct indication.	The signal is normally closed when in auto mode.  Need to confirm whether or not there will be a Transfer Inhibit Alarm.
CCO1 (A/M Status to the B/U CPU) Fails Open	None.	Assuming the Main CPU is in control and the controller is in auto, the operation will not be affected. There will be no Transfer Inhibit Alarm.	The signal is normally closed when in auto mode.  Need to confirm whether or not there will be a Transfer Inhibit Alarm.



FMEA of FWP Controller (FIC-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Digital Outputs Fail Closed</u>			
CCO0 (A/M Status to the Main CPU) Fails Closed	None.	The system operation is not affected unless other failures occur.	The signal is normally closed when in auto mode.
CCO1 (A/M Status to the B/U CPU) Fails Closed	None.	If the Main CPU is in control, and the controller is in auto, then the system operation is not affected.	The signal is normally closed when the controller is in auto.
		If the B/U CPU is in control, and the operator changes the controller to manual, the B/U CPU will not be able to detect it. The B/U CPU continues sending its FWP demand to the controller, until the deviation between the FWP demand calculated by the B/U CPU and the FWP controller output exceeds the setpoint, when the B/U CPU will fail and the FWP controller will transfer to manual.	

FMEA of FWP Controller (FIC-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of Power to Controller</u>			
Loss of power	The FWP controller will be off.	<p>All analog outputs fail to 0.  All digital outputs fail to Open status.  The failed signal will be sent to the Lovejoy FWP speed controller which will detect the failure and maintain the FWP speed at pre-failure value.</p> <p>The failed signal is sent to the CPUs for tracking, and after a delay will cause the CPUs to be failed due to deviation logic. As a result, the MFV, BFV and FWP controllers will transfer to manual control.</p>	Need to confirm operation of the Lovejoy controller.

**Table B.2-7 FMEA of Pressure Differential Indicating (PDI) Controller (PDI-4516/4517)**

The PDI controller is assumed to be in normal mode initially.

FMEA of PDI controller (PDI-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of analog input (Fail to 0.0 VDC)</u>			
ANI0 (Feed Regulating Valve Differential Pressure): Fail to 0.0 VDC	No alarms will be activated. The MFRV D/P bargraph will indicate D/P at 0.0 PSID.	MFRV differential pressure fail to 0.0 PSID. Operation of the DFWCS is not affected.	The signal is for display only.

FMEA of PDI controller (PDI-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
ANI1 (Main FRV Tracking Signal): Fail to 0.0 VDC	High rate deviation flags will be raised on the PDI controller and reset after a preset time period.	<p>A failed MFRV tracking signal is detected either because the current ANI1 signal is less than -20% or because its change rate is too high for the 0.0V DC input of the PDI ANI1. Upon this detection, the PDI controller thinks that the MFV fails although that ANI1 fails to zero does not mean the failure of MFV. The PDI controller will automatically take over by raising its output to the pre-failure value of the MFV output and enters the manual mode.</p> <p>If only the PDI ANI1 fails, the outputs of the normally running MFV and the PDI controllers will be summed together and should be twice as large as the output of the MFV controller. This will cause the MFRV to open more than designated by the CPU and the problem persists without operator's intervention. Plant analysis indicates that it will likely result in a failed open MFRV and transient. Without operator's action, the MFV demand deviation logic in the CPU software will fail the main CPU. After the B/U CPU takes over, the B/U CPU will fail for the same reason. Pump speed demand on another SG will be affected by the summed signal.</p>	<p>If the PDI ANI1 is the only failure, the operator may place the handswitch (HS) in the MFV Fail position. This will block the MFV output and only the PDI output will be sent to the MFRV. MFRV will be manually controlled by the operator via PDI controller.</p> <p>If, in addition to the PDI ANI1 failure, the MFV ANO0 demand output also fails to zero, the PDI controller will raise its output to the pre-failure MFV output. It is expected that the transfer from the MFV controller to the PDI controller is bumpless in this case.</p> <p>The MFV demand signal to the MFRV will be used by another S/G to calculate the pump speed demand.</p>

FMEA of PDI controller (PDI-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
ANI2 (Bypass FRV Tracking Signal): Fail to 0.0 VDC	No alarms are generated.	In high power control mode, the BFV controller should output linear 0% demand to the BFRV such that the BFRV is closed. Thus, the ANI2 Fail to 0.0VDC cannot be detected by comparing the failed ANI2 to the previous value held in the circular buffer of the PDI controller. The operation of the DFWCS will not be affected.	In low power control mode, a failed ANI2 is assumed to be detected by comparing the current ANI2 signal to the previously sampled values held in a circular buffer. PDI controller will not take over the BFV controller unless the manual switch HS-4516C/4517C is placed in the position of BFV Fail. Thus, the BFV will be continuously running as normal (output a linear 0% demand to the BFRV) and the operation of the DFWCS will not be affected. However, if the operator mistakenly decides to switch to the BFV Fail position, the summed outputs of the normally running BFV controller and the PDI (pre-failure value of the BFV controller) will open the BFRV wider than designated. Without operator's further action, the BFV demand deviation logic in the CPU software might fail the main CPU depending on the deviation setpoint. After the B/U CPU takes over, the B/U CPU might fail for the same reason.
<u>Loss of analog output (Fail to 0.0 VDC)</u>			

FMEA of PDI controller (PDI-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<p>ANO0 (Output to the MFRV or the BFRV): Fail to 0.0 VDC</p>	<p>During normal operation of the PDI, there is no alarm.</p>	<p>If neither the MFV controller nor the BFV controller fails, this has no effect on the system.</p> <p>If this failure occurs after the PDI controller takes over the MFV controller, the MFRV is expected to fail shut causing a loss of feedwater to the corresponding S/G.</p> <p>In high power mode, the BFRV is normally shut. Thus, if this failure occurs after the operator switches from the BFV controller to the PDI controller, no impacts are expected.</p>	<p>In low power mode, if this failure occurs after the PDI controller takes over the BFV controller, the BFRV is expected to fail shut causing a loss of feedwater to the corresponding S/G.</p>

FMEA of PDI controller (PDI-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Digital Inputs Fail Open</u>			
CCI0 (MFV Control Station Fail Flag - HS): Fail Open	No alarm is generated regarding this failure.	<p>The PDI controller will not be able to know whether HS-4516C/4517C have been placed in the MFV Fail position. If only CCI0 fails and the HS is in the normal or BFV Fail position, then the DFWCS operation is not affected since both MFV and/or BFV controllers are running as usual.</p> <p>If, in addition to CCI0 Fail Open, the MFV also fails, the PDI controller can still detect the MFV failure by comparing ANI1 signal to its previous values held in the circular buffer and automatically takes over the MFV controller.</p> <p>If, in addition to the CCI0 Fail Open, the operator thinks that the MFV controller has a problem even though the MFV demand output does not fail to zero and the rate change of the MFV demand output is not high, and decides to manually switch to the PDI controller, the PDI controller is not able to take over the MFV controller and the MFRV will fail shut. It is not certain about the response of the CPUs.</p>	<p>CCI0 Open=MFV OK and Closed=MFV Fail.</p> <p>The state of input CCI0 is decided by the position of HS-4516C/4517C. If the operator places the HS in the MFV Fail position, the output of MFV controller will be blocked and the output of the PDI controller will be sent to the MFRV.</p>

FMEA of PDI controller (PDI-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCIO (MFV Control Station Fail Flag - HS): Fail Closed	The PDI controller will display a message indicating that the MFV controller is failed although the MFV controller is not.	<p>If only CCIO fails, then the PDI controller will take over the MFV controller while the MFV is normally running. The output from PDI and the output from the MFV will be added together and sent to the MFRV, which will cause the MFRV to open more than designated by the CPU or the MFV controller. The operator must place HS-4516C/4517C to the Main Fail position in order to clear other contacts on the HS so that manual control of the MFRV using the PDI controller is obtained.</p> <p>Transients will be expected and instability may even be observed without this operator's action. Without operator's action, the MFV demand deviation logic in the CPU software will fail the main CPU depending on the deviation setpoint. After the B/U CPU takes over, the B/U CPU will fail for the same reason. The summed signal will be sent to another S/G to calculate the pump speed demand. The speed demand of the other SG will be affected. If both CCIO and the MFV controller fail, the PDI controller will automatically take over the MFV controller bumplessly.</p>	<p>CCIO Open=MFV OK and Closed=MFV Fail.</p> <p>S/G level can only be maintained by the operator's action in this situation.</p>



FMEA of PDI controller (PDI-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCI1 (BFV Control Station Fail Flag - HS): Fail Open	No alarms will be generated.	<p>The PDI controller will be unable to know whether HS-4516C/4517C have been placed in the BFV Fail position. If only CCI1 fails open and the HS is placed in the Normal or MFV Fail position, then the DFWCS operation is not affected since both MFV and/or BFV controllers are running as normal.</p> <p>If, in addition to the CCI1 Fail Open, the BFV also fails, and the operator placed the HS in the BFV Fail position, the output signal to the BFRV is the sum of the BFV output of linear 0% and the PDI output of linear -17%. BFRV might slight open.</p> <p>If, in addition to the CCI1 Fail Open, the BFV also fails, and the operator does not place the HS in the BFV Fail position, the operation of the system is not affected.</p> <p>If, in addition to the CCI1 Fail Open, the MFV controller also fails, the operation of the DFWS system is still not affected since the PDI controller will still take over the MFV controller automatically.</p>	<p>CCI1 Open=BFV OK and CCI1 Closed=BFV Failed</p> <p>The state of input CCI1 is determined by the position of HS-4516C/4517C.</p>

FMEA of PDI controller (PDI-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCI1 (BFV Control Station Fail Flag - HS): Fail Closed	The PDI controller will display a message indicating that the BFV controller is failed although the BFV controller is not.	<p>The CCI1 Fail Closed will make the PDI believe that BFV has failed and the PDI controller should raise the PDI's output to the pre-failure value of the BFV output. However, whether the output of the PDI controller should join the output of the MFV or the BFV is determined by the HS position, which is still at Normal position if the operator has not changed the position of the HS. Therefore, the output of the PDI controller will add to the output of the MFV.</p> <p>Because the BFRV is normally shut in high power control mode, the pre-failure value of the BFV controller held in the circular buffer of the PDI controller should be very small. The impacts of the summed signal on the MFRV may not be significant. Operators action that puts the HS at Bypass Fail position will regain the BFRV control via PDI.</p> <p>Without operator's action, whether the deviation logic will fail the controlling CPU depends on the deviation setpoint of the MFV demand although the deviation is small.</p>	The failure effects in lower power control mode is discussed in plant analysis. This will cause the MFRV to move to the open position and feedwater flow to the affected S/G will increase rapidly. The operator must place HS-4516C/4517C to the Bypass Fail position in order to regain control of the MFRV. This failure mode creates an overfeed situation for the affected S/G. Operator action is required in order to prevent overcooling of the RCS.

FMEA of PDI controller (PDI-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCI2 (Time Sync Input): Fail Open	Inconsistent time may be noticed by comparing operator's clock (independent clock in the control room or even the wristwatch), which was used to define the time stamp table, to the controller clocks.	CCI2 will be sampled periodically. If the CCI2 is closed, the PDI clock will be updated using the pre-defined time stamp. In case of the CCI2 Fail Open, the real-time clock of the PDI controller will not be updated. The clock values of the PDI controller will be propagated to other device controllers for time synchronization via the Microlink every minute.  As long as the Microlink is working correctly, a loss of synchronization between the device controllers will not happen. However, the times of device controllers are expected to be inconsistent with operator's clock. The synchronized times at individual controllers are not used in the control task but for the purpose of display only.	CCI2 Open=OK, do not update the PDI clock and CCI2 Closed=Sync, i.e., update the PDI clock.  It is assumed that updating the real-time clock of the PDI controller is performed when the system starts running. However, the operator is able to update the PDI clock at any time.  Updating the clock of the PDI controller can be either done manually or automatically. Automatic updating is not discussed in the available documentation.
CCI2 (Time Sync Input): Fail Closed	The time associated with the display does not change.	Real-time clock of the PDI controller will be updated using the same user-defined time-stamp table every cycle after sampling the CCI2.  If the time-stamp is not changed (which is assumed to be the case here), the time on the PDI (and then the times on other controllers) will remain the same.	CCI2 Open=OK, do not update the PDI clock and CCI2 Closed=Sync, i.e., update the PDI clock.
<u>Digital Outputs Fail Open</u>			

FMEA of PDI controller (PDI-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
CCO3 (Loss of Communication Alarm): Fail Open	No alarms are generated.	When CCO3 fails open, if Microlink is working properly, there will be no impact. If the Microlink fails, the loss of communication alarm will not be sent out and the plant computer will not be able to actuate the loss of communication alarm.	CCO3 Open=OK, i.e., the communication is normal. CCO3 Closed=A loss of communications alarm is actuated in the plant computer.
CCO3 (Loss of Communication Alarm): Fail Closed	A loss of communications signal will be sent to the DFWCS Trouble Alarm on the plant computer.	False alarm of a loss of communication will be sent to the plant computer if the Microlink is working properly. The alarm will persist until the failure is fixed.	Impacts on the operation of the DFWCS are not expected upon the failure of CCO3.

FMEA of PDI controller (PDI-4516/4517)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of Power to Controller</u>			
Loss of power that causes:  1. ANO0 fails to 0.0 mADC ; 2. CCO3 fails open; 3. Loss of the PDI controller	No alarms are generated for ANO0 Fail to 0.0 mADC.	<p>If neither the MFV controller nor the BFV controller fails, this has no effect on the system.</p> <p>If this failure occurs after the PDI controller takes over the MFV controller, the MFRV is expected to fail closed causing a loss of feedwater to the corresponding SG.</p> <p>In high power mode, the BFRV is normally shut. Thus, if this failure occurs after the operator switches from the BFV controller to the PDI controller, no impacts are expected.</p>	
	No alarms are generated for CCO3 Fail Open.	When CCO3 fails open, if the Microlink is working properly, there will be no impact. If the Microlink fails, the loss of communication alarm will not be sent out and the plant computer will not be able to actuate the alarm.	
	MFRV dP is no longer displayed on the PDI controller.	PDI fails its functions (display MFRV dP, detect failed MFV and BFV and change modes to take over manually or automatically). If other device controllers are working, this has no impact on the operation of DFWCS except for a loss of communication. Time synchronization will not be performed over the device controllers and FIX numbers from other three device controllers cannot be obtained.	

**Table B.2-8 FMEA of Optical Isolator (PB4R)**

PB4R is an optical isolator which performs conversions between electrical signal and optical signal and isolates the electrical coupling between inputs and outputs. Inputs pass through this isolator device and become the outputs. Thus, FMEA of inputs and the corresponding outputs are the same. Failure analysis of PB4R signals is not considered in plant analysis.

FMEA of Optical Isolator (PB4R)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 1 (B/U CPU Watchdog Timer Signal): Fail Closed	No alarms are generated.	The operation of the systems is not affected.  If, in addition to this failure, the B/U CPU truly fails in a way such that it can not send out this toggling watchdog timer signal, e.g., it gets hung, and this signal will remain low (contact fails closed), the watchdog timer will never timeout. If the B/U CPU is in control, a failover to the Main CPU will not occur and the system might lose automatic control.	The watchdog timer signal from the B/U CPU toggles every cycle. If the watchdog timer receives the low signal (contact becomes closed) within a preset time period, there will be no timeout, i.e., it is considered that the B/U CPU is working properly. Otherwise (contact becomes open), the watchdog timer will timeout and signal three device controllers.
Channel 1 (B/U CPU Watchdog Timer): Fail Open	Failover alarm will be displayed on the PDU and the B/U CPU failure will be alarmed via annunciator.	The operation of the system will not be affected since the Main CPU is in control. Watchdog timer of the B/U CPU will timeout and initiate a failure of the B/U CPU. The B/U CPU failure status will be sent to the controllers from the watchdog timer.  If, in addition to this failure, the Main CPU fails, a failover to the B/U CPU will not occur and the system has to be controlled manually.	The watchdog timer signal from the B/U CPU toggles every cycle. If the watchdog timer receives the low signal (contact becomes closed) within a preset time period, there will be no timeout, i.e., it is considered that the B/U CPU is working properly. Otherwise (contact becomes open), the watchdog timer will timeout and signal three device controllers.

FMEA of Optical Isolator (PB4R)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 2 (Main CPU Watchdog Timer): Fail Closed	No alarms are generated.	The operation of the systems is not affected.  If, in addition to this failure, the Main CPU truly fails in a way such that it cannot send out this toggling watchdog timer signal, e.g., it gets hung, this signal will remain low (contact fails closed) and the watchdog timer will never timeout. A failover to the B/U CPU will not occur and the system might lose automatic control.	The watchdog timer signal from the Main CPU toggles every cycle. If the watchdog timer receives the low signal (contact becomes closed) within a preset time period, there will be no timeout, i.e., it is considered that the Main CPU is working properly. Otherwise (contact becomes open), the watchdog timer will timeout and signal three device controllers.
Channel 2 (Main CPU Watchdog Timer): Fail Open	Failover alarm will be displayed on the PDU and the Main CPU failure will be alarmed via annunciator.	Watchdog timer of the Main CPU will timeout and initiate a failure of the Main CPU. The Main CPU failure status will be sent to the controllers from the watchdog timer. A failover to the B/U CPU will occur although the Main CPU is actually working properly.	The watchdog timer signal from the Main CPU toggles every cycle. If the watchdog timer receives the low signal (contact becomes closed) within a preset time period, there will be no timeout, i.e., it is considered that the Main CPU is working properly. Otherwise (contact becomes open), the watchdog timer will timeout and signal three device controllers.
Channel 3 (One Microprocessor Failed Signal): Fail Closed	See CCO3 Fail Closed in BFV FMEA	See CCO3 Fail Closed in BFV FMEA	Open=No microprocessor failed Closed=One microprocessor failed
Channel 3 (One Microprocessor Failed Signal): Fail Open	See CCO3 Fail Open in BFV FMEA	See CCO3 Fail Open in BFV FMEA	Open=No microprocessor failed Closed=One microprocessor failed
Channel 4 (Both Microprocessor Failed Signal): Fail Closed	See CCO2 Fail Closed in BFV FMEA	See CCO2 Fail Closed in BFV FMEA	Open=Not both microprocessors failed Closed=Both microprocessors failed

FMEA of Optical Isolator (PB4R)			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Channel 4 (Both Microprocessor Failed Signal): Fail Open	See CCO2 Fail Open in BFV FMEA	See CCO2 Fail Open in BFV FMEA	Open=Not both microprocessors failed Closed=Both microprocessors failed



### Appendix B.3 FMEA at Level of Major-Component-of-Module of DFWCS

The detailed FMEA at the level of components of the Main CPU module is shown in Table B.3.1. In the FMEA of the Main CPU module, the Main CPU module was decomposed into individual digital components, which were identified in Chapter 5. FMEA of each component was then conducted to determine the failure impacts on the component, the detectability of the failure, and the associated effects on the Main CPU module, i.e., failure modes of the Main CPU module. The failure rates of the components were then estimated using the generic data estimated in Chapter 9. The failure rates of the components were mostly estimated using a Hierarchical Bayesian method with raw data extracted from PRISM database [RAC]. The failure rates of different component failure modes were estimated using the failure mode distributions found mainly in two sources [Meeldijk 1996] and [RAC 1997b], as shown below. It should be noted that the column of “Detection of Failure Mode” in Table B.3-1 indicates the detection by the watchdog timer and software only. Table B.3-2 is the FMEA of a controller at a similar level of detail.

Failure mode distribution [Meeldijk 1996, RAC 1997b] of major components inside the Main CPU module is shown here:

1. Processor of the Main CPU:  
The failure mode “wrong data word” accounting for 60% of the total failure and the processor stops updating output upon the rest of the failures [RAC 1997b];
2. Associated components of the processor such as ISA bus, RAM, ROM (BIOS), flash disk, and buffer:  
Only one failure mode is assumed, i.e., loss of the components;
3. Address logic:  
It is sometimes called decoder and the failure mode distribution is: 40% of stuck high, 40% of stuck low, and 20% loss of logic (failure mode distribution for a typical digital component [Meeldijk 1996]);
4. Multiplexer and demultiplexer:  
Failure modes are defined in [Aeroflex 2005]. Note that each input of multiplexer corresponds to a sensor input and each output of demultiplexer corresponds to an analog output;
5. A/D and D/A Converter:  
Both A/D and D/A converters are linear IC circuits. The failure mode distribution is defined in [Meeldijk 1996]: 50% of degraded/improper output, 41% of no output, 3% of short circuit, 2% of open circuit, and 2% drift. There is only one A/D converter and one D/A converter. They are shared by all analog inputs and outputs;
6. Current Loop:  
Current loop is a linear device and the failure mode distribution is: 2% of fail-high, 44% of fail-low, and 52% of drifted output [Meeldijk 1996];
7. Digital output module:  
Digital signal output is implemented using solid-state switch. Status of output is controlled by opening or closing the switch. Its failure mode distribution is: 66.7% of fail to operate (fail as is) and 33.3% of false operation [RAC 1997b];
8. Digital input module:  
Digital signal input is also implemented using solid-state switch. See failure mode distribution for digital output module.
9. Application software:  
Data of application software failures are currently unavailable and need to be developed in the future.

**Table B.3-1 FMEA at Level of Components of DFWCS Modules: Main CPU**

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Application Software			
The application software on the main CPU seems to be normally running but sends erroneous output	No detection	Undetected Failure of Main CPU	1. Failure rate of the application software is the rate of occurrence of EFC. Further investigation is needed to determine it.
CPU is hung (CPU stops updating output)	Can be potentially detected by WDT if the WDT status is good.	Main CPU Fails to Send WDT the Toggling Signal	1. The WDT does not receive toggling signal and will trip the main CPU if the status of the WDT is normal.
Microprocessor of the Main CPU			
The CPU seems to be normally running but sends erroneous output (60% of total failure)	No detection	Undetected Failure of Main CPU	1. The CPU failure data is taken from Chapter 9. The failure rate is 3.3E-08 per hour. 2. The failure mode distribution used here is from [RAC 1997b]. This distribution is shows that a failure of “wrong data word” of a 16-bit CPU accounts for 60% of total failure. However, the Intel 80586 is a 32-bit processor. 3. Another failure mode distribution data from [Meeldijk 1996] can be used to replace the above failure mode distribution. The failure distribution in [Meeldijk 1996] shows that: stuck high or low accounts for 80% of the failure (this may correspond to the CPU stops updating outputs) and 20% of loss of logic (this may correspond to seemingly normal operation of the CPU).

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
CPU stops updating output (40% of the total failure)	Can be potentially detected by WDT if the WDT status is good.	Main CPU Fails to Send WDT the Toggling Signal	1. The WDT does not receive toggling signal and will trip the main CPU if the WDT is fine. This may correspond to other failures of CPU
ISA Bus			
Loss of ISA bus	Can be potentially detected by both application software and WDT if the WDT status is good.	Main CPU Fails to Send WDT the Toggling Signal	<p>1. Input and output of the CPU rely on the ISA bus and both the application software and the WDT can potentially detect this loss of the ISA bus. It is assumed the CPU is failed by the WDT if its status is normal.</p> <p>2. The failure rate of the bus is the sum of failure rates of line bus driver (4.6E-07 per hour) and receiver (6.2E-08 per hour) that are shown in Chapter 9. They are considered major components of the bus.</p>
RAM			
Loss of RAM	Can be potentially detected by WDT if the WDT status is good.	Main CPU Fails to Send WDT the Toggling Signal	<p>1. Application software has to be loaded into RAM to in order to run it. Thus, the application software can not run upon a malfunction of RAM. It is assumed that WDT can detect it because the Main CPU does not send out toggling signal any more.</p> <p>2. The failure rate (3.3E-07 per hour) is taken from Chapter 9.</p>
ROM (BIOS)			

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Loss of BIOS	Can be potentially detected by both application software and WDT if the WDT status is good.	Main CPU Fails to Send WDT the Toggling Signal	1. Input and output operation of CPU rely on BIOS routines. Both the software and the WDT can potentially detect this failure. It is likely that CPU will be failed by the WDT. 2. Failure rate (4.0E-08 per hour) is from Chapter 9, the failure rate of generic ROM.
Flash Disk			
Loss of Flash Disk	Can be detected by application software.	Main CPU Failed by Application Software (Needs further investigation)	1. Failure data of flash disk are unavailable in PRISM.
Serial Port			
Loss of Serial Port	No detection.	Main CPU Continues Normal Operation (Needs further investigation)	1. Serial port is used for communication between the Main CPU and PDU. Very likely the serial port is a RS-232 implementation. 2. The failure data is from PRISM for serial communication controller, the major component of serial communication port.
Multiplexer			
Loss of all signals	Can be detected by application software.	Main CPU Failed by Application Software	1. Deviation logic will capture the loss of input signals. Failure data are from [Aeroflex 2005]. 2. Only a brief description of failure effects of individual input signal though the multiplexer is shown here. Details of FMEA can be found in Appendix B.2.

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Loss of one of the signals: S/G 12 Feedwater Temperature	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 4 of Analog Backplane A and the signal is only used during low power operation.</li> <li>2. Invalidity of the signal will be detected by the Main CPU but the other signal is used and it has no effects on operation.</li> <li>3. A deviation alarm will be sent to plant computer from the Main CPU.</li> </ol>
Loss of one of the signals: S/G 11 Feedwater Temperature	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 5 of Analog Backplane A and the signal is only used during low power operation.</li> <li>2. Invalidity of the signal will be detected by the Main CPU but the other signal is used and it has no effects on operation.</li> <li>3. A deviation alarm will be sent to plant computer from the Main CPU.</li> </ol>
Loss of one of the signals: S/G 11 FWP A Bias	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 6 of Analog Backplane A.</li> <li>2. It will be detected by the Main CPU. The pump demand will be sent to the FWP regardless.</li> <li>3. A deviation alarm will be sent to the plant computer from the Main CPU.</li> </ol>
Loss of one of the signals: S/G 12 MFV Tracking	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 7 of Analog Backplane A.</li> <li>2. Higher MFV tracking signals from both S/Gs will be used to calculate FWP demand. Therefore, this loss of the signal does not affect the FWP demand calculation.</li> <li>3. There is no direct indication of the failure.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Loss of one of the signals: S/G 12 FWP A Tracking	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 8 of Analog Backplane A.</li> <li>2. A deviation larger than the setpoint between the CPU and the controller will cause a failover. If the deviation is not large enough, there is no effect. Here, we assume that the deviation is large.</li> <li>3. There is no direct indication of failure. If the Main CPU is failed, there will be an alarm to the plant computer.</li> </ol>
Loss of one of the signals: MFRV LVDT #2	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 13 of Analog Backplane A.</li> <li>2. If the accumulation exceeded the MFV-ACCUMULATION setpoint and the Diagnostic Transfer mode is enabled, the opposite positioner will be put in service and the control mode will be shifted to LOCKOUT.</li> <li>3. PDU and the associated CPU deviation annunciator will be activated.</li> </ol>
Loss of one of the signals: MFRV LVDT #1	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 14 of Analog Backplane A.</li> <li>2. If the accumulation exceeded the MFV-ACCUMULATION setpoint and the Diagnostic Transfer mode is enabled, the opposite positioner will be put in service and the control mode will be shifted to LOCKOUT.</li> <li>3. PDU and the associated CPU deviation annunciator will be activated.</li> </ol>
Loss of one of the signals: MFRV Differential Pressure #2	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 16 of Analog Backplane A. It appears that a loss of this signal does not affect the Main CPU's operation.</li> <li>2. Gooseneck purge related.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Loss of one of the signals: MFRV Differential Pressure #1	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 16 of Analog Backplane A. It appears that a loss of this signal does not affect the Main CPU's operation.</li> <li>2. Gooseneck purge related.</li> </ol>
Loss of one of the signals: S/G 11 Level #1	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 6 of Analog Backplane B.</li> <li>2. The other input is used for control.</li> <li>3. Failover will be displayed on PDU.</li> <li>4. If both S/G 11 Level signals are lost, there will be a loss of auto control.</li> <li>5. A deviation alarm and failover (if any) will be displayed on PDU.</li> </ol>
Loss of one of the signals: S/G 11 Level #2	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 7 of Analog Backplane B.</li> <li>2. The other input is used for control.</li> <li>3. Failover will be displayed on PDU.</li> <li>4. If both S/G 11 Level signals are lost, there will be a loss of auto control.</li> <li>5. A deviation alarm and failover (if any) will be displayed on PDU.</li> </ol>
Loss of one of the signals: S/G 11 FW Flow #1	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 8 of Analog Backplane B.</li> <li>2. The other input is used for control.</li> <li>3. Failover will be displayed on PDU.</li> <li>4. If both S/G 11 FW flow signals are lost, a single element control (high power mode) is adopted. Note that the Main CPU is conducting the single element control. If it is in low power mode, Low to High transfer is inhibited.</li> <li>5. A deviation alarm and failover (if any) will be displayed on PDU.</li> </ol>

FMEA of Major Components of Main CPU Module

Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Loss of one of the signals: S/G 11 FW Flow #2	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 9 of Analog Backplane B.</li> <li>2. The other input is used for control.</li> <li>3. Failover will be displayed on PDU.</li> <li>4. If both S/G 11 FW flow signals are lost, a single element control (in high power mode) is adopted. Note that the Main CPU is conducting the single element control. If it is in low power mode, Low to High transfer is inhibited.</li> <li>5. A deviation alarm and failover (if any) will be displayed on PDU.</li> </ol>
Loss of one of the signals: S/G 11 Main Steam Flow	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 10 of Analog Backplane B.</li> <li>2. The other input is used for control.</li> <li>3. Failover will be displayed on PDU.</li> <li>4. If both S/G 11 main steam flow signals are lost, a single element control (in high power mode) is adopted. Note that the Main CPU is conducting the single element control. If it is in low power mode, Low to High transfer is inhibited.</li> <li>5. A deviation alarm and failover (if any) will be displayed on PDU.</li> </ol>



FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Loss of one of the signals: S/G 12 Main Steam Flow	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 11 of Analog Backplane B.</li> <li>2. The other input is used for control.</li> <li>3. Failover will be displayed on PDU.</li> <li>4. If both S/G 11 main steam flow signals are lost, a single element control (in high power mode) is adopted. Note that the Main CPU is conducting the single element control. If it is in low power mode, Low to High transfer is inhibited.</li> <li>5. A deviation alarm and failover (if any) will be displayed on PDU.</li> </ol>
Loss of one of the signals: Neutron Flux #1	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 12 of Analog Backplane B.</li> <li>2. The other input will be used and control continues.</li> <li>3. If both inputs are lost, mode transfer is inhibited.</li> <li>4. A deviation alarm will be sent to plant computer.</li> </ol>
Loss of one of the signals: Neutron Flux #2	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 13 of Analog Backplane B.</li> <li>2. The other input will be used and control continues.</li> <li>3. If both inputs are lost, mode transfer is inhibited.</li> <li>4. A deviation alarm will be sent to plant computer.</li> </ol>
Loss of one of the signals: S/G 11 Level Setpoint	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 14 of Analog Backplane B.</li> <li>2. A deviation between this signal and the setpoint inside the program will occur. If it is larger than a fixed value, the internal level setpoint will be used. Otherwise, there is not impact.</li> <li>3. A deviation alarm will be sent to plant computer.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Loss of one of the signals: S/G 11 BFRV Tracking	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 15 of Analog Backplane B.</li> <li>2. Control continues and BFRV will be closed. There is no impact when it is in high power mode. If it is in low power mode, a failover will occur.</li> <li>3. There is no alarm.</li> </ol>
Loss of one of the signals: S/G 11 MFRV Tracking	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 16 of Analog Backplane B.</li> <li>2. The deviation between the Main CPU output and controller feedback will cause a failover for a large deviation. The large deviation is assumed to be the case here. If the deviation is small, the control continues.</li> <li>3. A deviation alarm and the failover (if any) will be displayed on PDU.</li> </ol>
A/D Converter			
All 16 bits stuck at zeros or ones (48% of the total failure)	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Both A/D and D/A converters are linear ICs.</li> <li>2. Failure data (2.44E-09 per hour) are PRISM raw data of 16-bit A/D or D/A converter. Therefore a Bayesian update might be necessary.</li> <li>3. The failure mode distribution is from [Meeldijk 1996]. The failure mode distribution of a linear IC is: 50% of degraded/improper output, 41% of no output, 3% of short circuit, 2% of open circuit, and 2% drift.</li> <li>4. Since the A/D shared by all inputs, loss of A/D results in a loss of all inputs.</li> </ol>
Random bit failure (52% of the total failure)	No detection.	Undetected Failure of Main CPU	<ol style="list-style-type: none"> <li>1. Although some of random failures might be detected by the application software, the failures are conservatively assumed to be undetectable.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
D/A Converter			
Output fails high (2% of the total failure)	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Main CPU will detect this failure via controller feedback if the status of the controller is normal..</li> <li>2. Failure data (2.44E-09 per hour) are from PRISM raw data.</li> <li>3. Failure mode distribution is from [Meeldijk 1996] (refer to comment 3 of A/D converter).</li> <li>4. Since the D/A shared by all inputs, loss of D/A results in a loss of all inputs.</li> </ol>
Output fails low (44% of total failure)	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. In addition to failure of the main CPU, PDI controller will take over the MFV controller</li> </ol>
Drifted output (52% of the total failure)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Output drifted within certain range can be coped with</li> </ol>
Demultiplexer			
Loss of all output signals	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. The Main CPU has three analog outputs: the controller demands.</li> <li>2. DEMUX is considered similar to MUX and the failure data are also from [Aeroflex 2005]. (also refer to MUX above).</li> <li>3. In addition to the failure of the Main CPU, PDI controller will take over the MFV controller for this failure mode.</li> <li>4. Only a brief description of failure effects of individual input signal though the multiplexer is shown here. Details of FMEA can be found in Appendix B.2.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Loss of one of the output signals: Feed Pump Demand	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 1 of Analog Backplane A.</li> <li>2. There is no direct indication of this failure. Main CPU deviation (between its demand output and FWP tracking signal) will be sent to plant computer.</li> <li>3. It seems Lovejoy controller will detect this failure and takes over but details are not available (Appendix B.2).</li> </ol>
Loss of one of the output signals: Bypass Valve Demand	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 2 of Analog Backplane A.</li> <li>2. The BFV demand signal is normally zero in high power mode. Nothing will happen for loss of the signal.</li> <li>3. There is no direct indication of this.</li> </ol>
Loss of one of the output signals: Main Valve Demand	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 3 of Analog backplane A.</li> <li>2. In addition to the failure of the Main CPU, PDI controller will take over the MFV controller for this failure mode.</li> <li>2. The PDI controller will display a “MFV fail” message. Main CPU will also activate a deviation message.</li> </ol>
Current Loop			

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Output current fails high (2% of the total failure): Feed Pump Demand	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 1 of Analog Backplane A.</li> <li>2. A failover will occur due to the large deviation between the CPU demand and the FWP tracking signal.</li> <li>3. There is no direct indication of this failure. Main CPU deviation (between its demand output and FWP tracking signal) will be sent to plant computer.</li> <li>4. It seems Lovejoy controller will detect this failure and takes over but details are not available (Appendix B.2).</li> <li>5. Current loop is a linear device. The failure rate is 2.43E-09 per hour from PRISM raw data of IC, Linear, Transmitter/receiver. The failure mode distribution is shown in [Meeldijk 1996].</li> <li>6. It is assumed there is a separate current loop for each output.</li> </ol>
Output current fails low (44% of the total failure): Feed Pump Demand	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 1 of Analog Backplane A.</li> <li>2. A failover will occur due to the large deviation between the CPU demand and the FWP tracking signal.</li> <li>3. There is no direct indication of this failure. Main CPU deviation (between its demand output and FWP tracking signal) will be sent to plant computer.</li> <li>4. It seems Lovejoy controller will detect this failure and takes over but details are not available (Appendix B.2).</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Drifted output current (52% of the total failure): Feed Pump Demand	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 1 of Analog Backplane A.</li> <li>2. According to Appendix B.2, this failure can be compensated by the control algorithm.</li> <li>3. There is not direction indication of this failure.</li> </ol>
Output current fails high (2% of the total failure): Bypass Valve Demand	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 2 of Analog Backplane A.</li> <li>2. According to Appendix B.2, the CPU deviation logic for the BFV demand signal is inhibited in high power mode. However, if the BFV demand increases, the MFV demand will decrease to cope with this. Therefore, there is at most a transient.</li> <li>3. There is no direct indication of this failure.</li> </ol>
Output current fails low (44% of the total failure): Bypass Valve Demand	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 2 of Analog Backplane A.</li> <li>2. The BFV demand signal is normally zero in high power mode. Nothing will happen for loss of the signal.</li> <li>3. There is no direct indication of this.</li> </ol>
Drifted output current (52% of the total failure): Bypass Valve Demand	Can be detected by application software.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 2 of Analog Backplane A.</li> <li>2. According to Appendix B.2, a proper setpoint can cope with this.</li> <li>3. There is no direct indication of this failure.</li> </ol>
Output current fails high (2% of the total failure): Main Valve Demand	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 3 of Analog Backplane A</li> <li>2. Main CPU will detect this failure via MFV controller feedback if the MFV controller status is normal.</li> <li>3. There is no direct indication of this failure. A deviation alarm will be sent to the plant computer from the Main CPU.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Output current fails low (44% of the total failure): Main Valve Demand	Can be detected by application software.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. PDI controller will take over the MFV controller.</li> <li>2. According to Appendix B.2, PDI will take over before the failure of the Main CPU.</li> <li>3. The PDI controller will display a “MFV fail” message. The Main CPU will give a deviation message.</li> </ol>
Drifted output current (52% of the total failure): Main Valve Demand	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. According to Appendix B.2, output drifted within certain range can be compensated.</li> <li>2. There is no direct indication of this failure.</li> </ol>
$V_{REF}$			
Loss of $V_{REF}$	No detection	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. According to plant information, <math>V_{REF}</math> is only used to correct for voltage offsets in the input signal path when the system is initialized.</li> </ol>
Analog Address Logic			
Unintended address sent out and wrong component selected (20% of the total failure)	Not likely to be detected by the application software and not detectable by the WDT.	Undetected Failure of Main CPU	<ol style="list-style-type: none"> <li>1. Although some of address logic failures might be detected by the application software, it is conservatively assumed to be undetectable.</li> <li>2. Address logic is usually called decoder in current digital systems. Failure data (7.0E-08 per hour) is from Chapter 9.</li> <li>3. An analog address logic is a digital device and the failure mode distribution is from [Meeldijk 1996]: 40% of stuck high, 40% of stuck low, and 20% loss of logic.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Completely loss of analog address logic (80% of the total failure)	Can be potentially detected by both application software and WDT if the WDT status is good.	Main CPU Fails to Send WDT the Toggling Signal	1. CPU should be able to detect the status of analog address logic but can not send out output properly.
Buffer			
Loss of buffer	Can be potentially detected by WDT if the WDT status is good.	Main CPU Fails to Send WDT the Toggling Signal	1. All digital input and output require the buffer. 2. The failure rate is from Chapter 9.
Digital Address Logic			
Unintended address sent out and wrong component selected	Not likely to be detected by the application software and not detectable by the WDT.	Undetected Failure of Main CPU	1. Although some of failures might be detected by the WDT, it is conservatively assumed to be undetectable. 2. Failure data and failure mode distribution are the same to analog address logic.
Complete loss of digital address logic	Can be detected by WDT if the WDT status is good.	Main CPU Fails to Send WDT the Toggling Signal	1. CPU should be able to detect the status of analog address logic but can not send digital output properly
Digital Output Module			



FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Failure to operate of the solid-state switch (Watchdog Timer fails as is)	Can be detected by WDT if the WDT status is good.	Main CPU Fails to Send WDT the Toggling Signal	<ol style="list-style-type: none"> <li>1. Channel 0 of Digital Backplane: output to WDT.</li> <li>2. Failure data is from PRISM and failure mode distribution is from [RAC 1997b].</li> <li>3. PDU and the plant computer should indicate the failure of the Main CPU.</li> <li>4. The main component of the digital output module is the solid-state switch. The failure mode distribution, according to [RAC 1997b], is 66.7% for Failure to Operate, and 33.3% for False Operation. The failure rate of digital switch from PRISM is 2.43E-09 per hour.</li> </ol>
Failure to operate of the solid-state switch (Power Fail as is)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 2 of Digital Backplane: power failure or the CPU is not controlling. Power failure fails as (normally not energized) it indicates that the Main CPU is OK. Therefore, this failure does not affect the operation of the Main CPU or the system until there is a power failure of the Main CPU. In that case, there will be a undetected Main CPU failure and a loss of auto control.</li> <li>2. There is no direct indication or detection of this failure.</li> </ol>
False operation of the solid-state switch (Power Fail fails to opposite state)	No detection.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. False operation of this switch will indicate that the Main CPU power failure and a fail-over should occur.</li> <li>2. There is no direct indication or detection of this failure. There should be indirect indication from the PDU and the plant computer.</li> </ol>

FMEA of Major Components of Main CPU Module

Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Failure to operate of the solid-state switch (High Power Indication fails closed)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 4 of Digital Backplane: high power indication. It is normally closed indicating the high power mode.</li> <li>2. This failure does not affect the Main CPU or the system operation. It might, however, affect operator since this failure indicates the high power mode but it is actually the low power mode.</li> <li>3. There is no direct indication of this failure.</li> </ol>
False operation of the solid-state switch (High Power Indication fails open)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. There is no direct indication of this failure.</li> <li>2. This failure does not affect the Main CPU or the system operation. It might, however, affect operator since this failure indicates the low power mode but it is actually the high power mode.</li> </ol>
False operation of the solid-state switch (Transfer Indication fails closed)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 5 of Digital Backplane: transfer indication. It is normally open indicating there is no mode transfer.</li> <li>2. There is no direct indication of this failure. This failure indicates that the system is transferring between power modes.</li> <li>3. This failure does not affect the Main CPU or the system operation. It might, however, affect operator since this failure indicates a undergoing transfer but there is actually no transfer.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Failure to operate of the solid-state switch (Transfer Indication fails open)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. There is no direct indication of this failure. This failure indicates there is no power mode transfer even a transfer is undergoing.</li> <li>2. This failure does not affect the Main CPU or the system operation. It might, however, affect operator since this failure indicates no transfer but there is actually one.</li> </ol>
False operation of the solid-state switch (Low Power Indication fails closed)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 6 of Digital Backplane: low power indication. It is normally open (high power mode). This failure indicate that the system is operating in high power mode.</li> <li>2. There is no direct indication of this failure.</li> <li>3. This failure does not affect the Main CPU or the system operation. It might, however, affect operator.</li> </ol>
Failure to operate of the solid-state switch (Low Power Indication fails open)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. There is no direct indication of this failure.</li> <li>2. This failure does not affect the Main CPU or the system operation. It might, however, affect operator.</li> </ol>
False operation of the solid-state switch (Bypass Override Indication fails closed)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 7 of Digital Backplane: bypass override (BPO) indication. It is normally open (not in BPO mode). This failure indicates that the system is in a BPO mode.</li> <li>2. There is no direct indication of this failure.</li> <li>3. This failure does not affect the Main CPU or the system operation. It might, however, affect operator.</li> </ol>
Failure to operate of the solid-state switch (Bypass Override fails open)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. There is no direct indication of this failure.</li> <li>2. This failure does not affect the Main CPU or the system operation. It might, however, affect operator.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
False operation of the solid-state switch (Deviation Alarm fails closed)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 8 of Digital Backplane: deviation alarm and it is normally open, i.e., there is no deviation. This failure indicates that there is a deviation. If this output is closed then there is a deviation.</li> <li>2. It seems that a fail-over will occur regardless of the state of this output.</li> <li>3. There is no direct indication. However, the plant computer will indicate that the Main CPU detects a deviation.</li> </ol>
Failure to operate of the solid-state switch (Deviation Alarm fails open)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This failure indicates there is no deviation even there is. It does not affect operation of the CPUs or the system.</li> <li>2. There is no direct indication.</li> </ol>
Failure to operate of the solid-state switch (Transfer Inhibit fails open)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 9 of Digital Backplane: transfer inhibit. It is normally open, i.e., transfer is not inhibited.</li> <li>2. This fails-open failure indicates that the transfer is not inhibited.</li> <li>3. Transfer is not considered in this study.</li> <li>4. There is no direct indication.</li> </ol>
False operation of the solid-state switch (Transfer Inhibit fails closed)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This fail-closed indicates that control mode transfer is inhibited.</li> <li>2. There is no direct indication. However, the plant computer will indicate that power modes transfer is inhibited.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Failure to operate of the solid-state switch (Positioner Selected fails closed)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 11 of Digital Backplane: positioner selected. Output to positioner.</li> <li>2. It is assumed here that position A is normally used, i.e., output is closed.</li> <li>3. This fail closed (fail as is) will not affect the operation of the Main CPU. However, if the accumulated deviation between the demand from the Main CPU and the position of the MFRV exceeds a setpoint value, e.g., the positioner A fails, the Main CPU can not switch to positioner B. It might lead to reactor trip.</li> </ol>
False operation of the solid-state switch (Positioner Selected fails open)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This fail open will not affect the operation of the Main CPU if the positioner B is in a good state. However, if the accumulated deviation between the demand from the Main CPU and the position of the MFRV exceeds a setpoint value, e.g., the positioner B fails, the Main CPU can not switch to positioner A. It might lead to reactor trip.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Failure to operate of the solid-state switch (No Failures in Microprocessor fails open)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 13 of Digital Backplane: no failures in microprocessor. It is assumed to be normally open, i.e., the Main CPU does not failure. This output goes to the other microprocessor.</li> <li>2. The failure indicates that the Main CPU is in a good state. This will not affect the operation of the Main CPU and the system. If the Main CPU truly fails, the Backup CPU will be able obtain the Main CPU's status directly from the Main CPU instead from this output. Thus, it seems that the failure of the Main CPU will not cause problems.</li> <li>3. PDU and plant computer will show the status of the Main CPU.</li> <li>4. There is no direct indication of this failure.</li> </ol>
False operation of the solid-state switch (No Failure in Microprocessor fails closed)	No detection.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. The failure signals that the Main CPU fails and effectively, the MFV controller will block the demand from the Main CPU. The control demand will be from the Backup CPU. Thus, if the Backup CPU is in a good state, it causes a fail-over only and will not affect the operation of the system.</li> <li>2. If, in addition to the Main CPU failure, the Backup CPU also fails, there will be a loss of auto control.</li> <li>3. PDU will show the status of the Main CPU.</li> <li>4. There is no direct indication of this failure. Failure status of the Main CPU will be displayed by the PDU.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Failure to operate of the solid-state switch (No Deviation fails open)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 14 of Digital Backplane: no deviations. Thus, it is normally open, i.e., there is no deviation. This output goes to the other CPU (the Backup CPU).</li> <li>2. The failure of fail-open (fails as is) indicates that there is no deviation. Thus, this failure does not affect the operation of the Main CPU or the system. However, if there is truly a deviation, the Backup CPU will not know due to this failure.</li> <li>3. Even though, the Backup CPU can receive the Main CPU failure (due to deviation) status from the MFV controller. Thus, it is still likely that the deviation will cause a fail-over.</li> <li>4. This is not indication of this failure. If there is a deviation, the PDU and the plant computer will show the message.</li> </ol>
False operation of the solid-state switch (No Deviation fails closed)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. The failure of fail-closed indicates that the Main CPU has a deviation. However, this failure does not cause the Main CPU to fail and thus, the Main CPU remains in control.</li> <li>2. Further investigation is needed.</li> <li>3. This is not indication of this failure. The status of the Main CPU will be "Failure" in the PDU display.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Failure to operate of the solid-state switch (CPU Level Status to the Other CPU fails open)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 15 of Digital Backplane: CPU level status. It is normally open indicating that both SG level signals are valid. This signal goes to the Backup CPU. Thus, the failure of fail-open (fail as is) indicates the validity of signals and will not cause any problem with the operation of the Main CPU and the system.</li> <li>2. If, in addition to this failure, both the signal level are invalid, the Main and the Backup CPUs will fail and there will be a loss of auto control.</li> <li>3. If, in addition to this failure, the Main CPU fails, it will inform the Backup CPU its status and the Backup CPU will takes over.</li> <li>4. There is no direct indication of this failure.</li> </ol>
False operation of the solid-state switch (CPU Level Status to the Other CPU fails closed)	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. The failure of fail-closed indicates that both SG level signals are invalid. Since the Main CPU is in a good state and the Backup CPU can validate the signals, it should not cause any problem.</li> <li>2. There is no direct indication of this failure.</li> </ol>
Failure to operate of the solid-state switch	N/A		This is the signal of feedflow/steamflow status to the Backup CPU.
False operation of the solid-state switch	N/A		
Digital Input Module			



FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
A/M Status BFV fails closed	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 16 of Digital Backplane: A/M Status BFV. It is normally closed, i.e., the BFV is in auto status. It is an input from the BFV controller.</li> <li>2. It does not cause any problem with the Main CPU or the system.</li> <li>3. If, in addition to this failure, the BFV controller is in manual mode (actually, this is already the failure of system because of a loss of auto control), the Main CPU would still think it is controlling. When the deviation is large, there will be a failover. However, the Backup CPU knows the BFV is in manual mode.</li> <li>4. The major component of digital input is again a solid-state switch [Eurotherm 2000]. Therefore, the failure rate and failure mode distribution are 2.43E-09 per hour and 66.7% of fail to operate and 33.3% for false operation.</li> <li>5. There is not direct indication of this failure.</li> </ol>
A/M Status BFV fails open	No detection.	Main CPU Tracking	<ol style="list-style-type: none"> <li>1. The failure indicates that the BFV is in manual status. The Main CPU would track instead of control and the BFRV may drift open. However, it will be compensated by the MFV.</li> <li>2. There is no indication of this failure. MFV status displayed on the PDU and the FWP controller is different.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
A/M Status MFV fails closed	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 17 of Digital Backplane: A/M Status MFV. It is normally closed, i.e., the MFV is in auto status. It is an input from the MFV controller.</li> <li>2. It does not affect the operation.</li> <li>3. If, in addition to this failure, the MFV controller is in manual mode (actually, this is already the failure of system because of a loss of auto control), the Main CPU would still think it is controlling. When the deviation is large, there will be a failover. However, the Backup CPU knows the MFV is in manual mode.</li> <li>4. There is not direct indication of this failure.</li> </ol>
A/M Status MFV fails open	No detection.	Main CPU Tracking	<ol style="list-style-type: none"> <li>1. The failure indicates that the MFV is in manual status and the Main CPU will track instead of control. The MFRV will drift from setpoint. Eventually, the system will fail without operator's actions.</li> <li>2. There is no indication of this failure. MFV status displayed on the PDU and the FWP controller is different.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
A/M Status FWP fails closed	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 18 of Digital Backplane: A/M Status FWP. It is normally closed, i.e., the FWP is in auto status. It is an input from the FWP controller.</li> <li>2. This failure indicates that the FWP is auto. Operation of the Main CPU or the system is not affected.</li> <li>3. If, in addition to this failure, the FWP controller is actually in manual status (actually, this situation is already a system failure due to loss of auto control), the Main CPU is still thinking it is controlling the FWP. When the deviation is large enough, there will be a failover. After the failover, the Backup CPU will know the correct status of the FWP controller.</li> <li>4. There is no direct indication of this failure.</li> </ol>
A/M Status FWP fails open	No detection.	Main CPU Tracking	<ol style="list-style-type: none"> <li>1. This failure indicates that the FWP controller is in manual status. The Main CPU will track instead of control. The pump demand may windup but it is expected to be compensated by the MFV controller.</li> <li>2. There is no direct indication of this failure. However, the FWP controller status displayed by the PDU and the FWP controller is different.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Reactor Trip fails closed	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 19 of Digital Backplane: Reactor Trip. It is normally closed, i.e., there is no reactor trip. It is an input from post reactor trip position relay.</li> <li>2. This failure does not affect the system. However, the Main CPU can not detect whether there is a reactor trip.</li> <li>3. If, in addition to this failure, there is a reactor trip, the Main CPU will fail and a failover will occur (Appendix B.2).</li> <li>4. There is no direct indication of this failure.</li> </ol>
Reactor Trip fails open	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This failure indicates that there is a reactor trip. Trip functions will be activated after certain time period (Appendix B.2).</li> <li>2. A reactor trip will occur.</li> </ol>
Main/Backup CPU Identification fails closed	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 20 of Digital Backplane: Main/Backup CPU Identification. It is normally closed, i.e., the pre-selected CPU is the Main CPU. This failure mode can not occur to the Main CPU (Appendix B.2). It is a pre-selected input.</li> <li>2. However, the failure will make the Backup CPU think that it is the Main CPU and starting controlling. The Backup CPU will fail due to deviation.</li> <li>3. There is no indication of this failure.</li> </ol>
Main/Backup CPU Identification fails open	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This failure can not occur to the Main CPU (Appendix B.2).</li> <li>2. It does not affect the Backup CPU.</li> <li>3. There is no indication of this failure.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Turbine Trip fails closed	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 21 of Digital Backplane: Turbine Trip. It is normally closed, i.e., there is no turbine trip. It is an input from the turbine relay.</li> <li>2. This failure does not affect the operation. The system can not detect the occurrence of turbine trip.</li> <li>3. If, in addition to this failure, there is a turbine trip, a reactor trip will follow and the system remains in automatic control.</li> <li>4. There is no indication of this failure.</li> </ol>
Turbine Trip fails open	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This indicates that there is a turbine trip.</li> <li>2. The MFRV will be shut down but the Main CPU remains in automatic control.</li> <li>3. There is not indication of this failure except a reactor trip. PDU will display the trip events.</li> </ol>
Main CPU Failed fails closed	No detection.	Main CPU Failed by Application Software	<ol style="list-style-type: none"> <li>1. Channel 22 of Digital Backplane: Main CPU Failed. It is normally open, i.e., the Main CPU is not failed. It is an input from the MFV controller.</li> <li>2. This failure indicates that the Main CPU is failed, a failover is expected.</li> <li>3. Main CPU failure will be displayed by the PDU and the plant computer.</li> </ol>
Main CPU Failed fails open	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This failure indicates that the Main CPU is OK even it is not. Thus, it does not affect the operation.</li> <li>2. If, in addition to this failure, the Main CPU fails, its true status can be detected by the MFV controller and it will be failed. The Backup CPU will take over.</li> <li>3. There is no indication of this failure.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Backup CPU Failed fails closed	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 23 of Digital Backplane: Backup CPU Failed. It is normally open, i.e., the Backup CPU is OK. It is an input from the MFV controller.</li> <li>2. This failure indicates that the Backup CPU failed. It does not affect the operation.</li> <li>3. If, in addition to this failure, the Backup CPU fails, it is still OK since the Main CPU is controlling.</li> <li>4. If, in addition to this failure, the Main CPU failed, there will be a failover because the MFV knows the true status of the Backup CPU.</li> <li>5. The PDU and the plant computer will show the failure status of the Backup CPU.</li> </ol>
Backup CPU Failed fails open	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This failure indicates that the Backup CPU is OK. It does not affect the operation.</li> <li>2. If, in addition to this failure, the Backup CPU failed, it is still OK.</li> <li>3. If, in addition to this failure, the Main CPU fails, there will be a failover to the Backup CPU.</li> <li>4. There is no indication of this failure.</li> </ol>
Time Sync	N/A	N/A	Not used. It is an input from the external clock.

FMEA of Major Components of Main CPU Module

Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Neutron Flux #1 Bypass fails close	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 25 of Digital Backplane: Neutron Flux #1 Bypass. It is normally closed, i.e., the flux signal is not bypassed. It is an input from the keyswitch.</li> <li>2. This failure indicates that the flux #1 is not bypassed. If the external keyswitch is “normal”, it does not affect the operation.</li> <li>3. However, even if the external keyswitch is “bypass”, it does not seem that the operation will be affected (Appendix B.2).</li> <li>4. There is no indication of this failure.</li> </ol>
Neutron Flux #1 Bypass fails open	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This failure indicates that the flux #1 is bypassed even if the external keyswitch is “normal”. It does not affect the operation of the system.</li> <li>2. There is no indication of this failure.</li> </ol>
Neutron Flux #2 Bypass fails closed	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 26 of Digital Backplane: Neutron Flux #2 Bypass. It is normally closed, i.e., the flux signal is not bypassed. It is an input from the keyswitch.</li> <li>2. This failure indicates that the flux #2 is not bypassed. If the external keyswitch is “normal”, it does not affect the operation.</li> <li>3. However, even if the external keyswitch is “bypass”, it does not seem that the operation will be affected (Appendix B.2).</li> <li>4. There is no indication of this failure</li> </ol>
Neutron Flux #2 Bypass fails open	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This failure indicates that the flux #2 is bypassed even if the external keyswitch is “normal”. It does not affect the operation of the system.</li> <li>2. There is no indication of this failure.</li> </ol>

FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
Positioner Selected fails closed	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 27 of Digital Backplane: Positioner Selected. It is normally closed, i.e., positioner A is selected. It is an input from the positioner.</li> <li>2. This failure indicates that the positioner A is selected as the active positioner. It does not affect the operation.</li> <li>3. There is no indication of this failure.</li> </ol>
Positioner Selected fails open	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This failure indicates that positioner B is the active positioner. It does not affect the operation.</li> <li>2. There is no direct indication of this failure. PDU will show the active positioner.</li> </ol>
No Failures in Other Microprocessor fails closed	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 28 of Digital Backplane: No Failures in Other Microprocessor. It is normally closed, i.e., the other microprocessor is not failed. It is an input from the other microprocessor.</li> <li>2. This failure indicates that the other microprocessor is OK. It does not affect the operation of the Main CPU.</li> <li>3. There is no indication of this failure.</li> </ol>
No Failures in Other Microprocessor fails open	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This failure indicates that the other microprocessor is failed. It does not affect the operation of the Main CPU.</li> <li>2. If, in addition to this failure, the Backup CPU failed, there will be no failover. A loss of automatic control occurs.</li> <li>3. There is no indication of this failure.</li> </ol>



FMEA of Major Components of Main CPU Module			
Failure Mode	Detection of Failure Mode	Failure Effects (In Terms of States of Main CPU)	Comments
No Deviation in Other Microprocessor fails closed	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 29 of Digital Backplane: No Deviations in Other Microprocessor. It is normally closed, i.e., there is no deviation in the other microprocessor. It is an input from the other CPU.</li> <li>2. This failure indicates that the other CPU is OK. It does not affect the operation.</li> <li>3. There is no indication of this failure.</li> </ol>
No Deviation in Other Microprocessor fails open	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This failure indicates that the other CPU has a deviation. It does not affect the operation.</li> <li>2. If, in addition to this failure, the Main CPU failed, the failover will not occur and there will be a loss of automatic control.</li> <li>3. There is no indication of this failure.</li> </ol>
Both Level Signals Valid in Other Microprocessor fails closed	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. Channel 30 of Digital Backplane: Both level signals are valid in the other microprocessor. It is normally open, i.e., both signals are valid.</li> <li>2. This failure indicates that level signals in the Backup CPU are invalid. It does not affect the operation of the Main CPU.</li> <li>3. There is no indication of this failure.</li> </ol>
Both Level Signals Valid in Other Microprocessor fails open	No detection.	Main CPU Continues Normal Operation	<ol style="list-style-type: none"> <li>1. This failure indicates that both level signals in the Backup CPU are valid. It does not affect the operation.</li> <li>2. There is no indication of this.</li> </ol>
Both Steam Flow and Both FW Flow Signals Valid in Other Microprocessor	N/A	N/A	Not used.

**Table B.3-2 FMEA of F&P 53MC5000 Controller**

FMEA of F&P 53MC5000			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
<u>Loss of Power Supplies</u>			
Loss of +15V Supply	Display goes blank	Loss of analog input and output channels due to loss of D/A and analog comparator powered by +15V. The RS-232 serial communication link is lost. Loss of power supply to a level shifter (from 5V to 12V digital signals) in the display circuit.	Loss of D/A has the same effects on the analog input and output channels.
Loss of -15V Supply	No indication of failure unless the configuration port is in use.	Loss of RS-232 serial communication link.	This failure has no effect on the controlled application (unless the port is used to receive control information which is usually not the case).
Loss of +26V	Not detectable unless the analog output is monitored using extra circuits.	Loss of analog outputs.	If analog outputs are monitored, then the failure of the monitoring circuits and/or the sampling and holding circuits has the same failure effects.
Loss of +5V	The display probably goes blank	Most of functions will be lost. The drive lines to the display will be lost. The display was designed to blank if the input data stream stops to protect the display. Analog outputs will drift.	Display interface design is not tested yet for this situation.
Loss of +80V	Display goes blank	Only display is affected.	
Loss of -110V	Display goes blank	Only display is affected.	

FMEA of F&P 53MC5000			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
ROM Error	The processor performs a checksum test of ROM at startup or reset. If the ROM fails at this time, it will be detected. Otherwise, it is not detectable.	When the processor is running, the failure effects of a ROM error is not predictable.	ROM is usually used to store programs and constants used in the programs developed by vendors.  There is no continuously executing memory error detection algorithm.
RAM Error	The processor performs a test of RAM at startup. If the RAM fails at this time, it will be detected. Otherwise, it is not detectable.	When the processor is running, the failure effects of a RAM error is not predictable.	For 53MC5000, there is no background running process that performs read/write test (however, 53MC2000 does) and there is no continuously executing memory error detection algorithm.
PAL (Programmable Array Logic) Error	Unknown	Some functions provided possibly by user-written F-TRAN software stored in RAM will not be available.	
Computational Errors	Unknown	Unknown except that CPU outputs will be incorrect.	Plant information indicates that the risk of computational problems caused by the math library is low.
Initialized Data Errors	Unknown	If the presence of wrong data is noticed before the operation, there will be no impacts. Otherwise, severity of impacts on the DFWCS depends on individual errors of data.	Re-initialization of the software should fix the non-default data errors since all non-default database values are hardcoded into the various software modules used in the controllers.

FMEA of F&P 53MC5000			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Loss of Lithium Battery	Error will be obvious only after the external power supply is also lost.	Loss of all functions of the processor due to a loss of program and database if the external power supply is unavailable. Otherwise, the processor continue to run.	
Loss of RS-485 Serial Communications Interface	N/A	53MC5000 will not be able to receive data upon problem with the receive circuit. 53MC5000 will not be able to transmit data upon problem with the transmit circuit.	From available documentation, RS-485 serial communication is not used in the DFWCS (RS-232 is used for development only).
Loss of RS-485 Jabber	A DFWCS trouble alarm will be actuated.	53MC5000 does not use the communication network to transmit control related information. The failure effects could be losses of warning messages or time.	It is assumed that RS-485 Jabber indicates the Microlink communication link.
Loss of PWR_ON Signal	Flashing display.	Watchdog time out due to loss of reset signal from PWR_ON. The processor will halt. The control task stops updating outputs and the display task stops updating display memory. All the contact outputs will be at "Open" state. Analog outputs will go to zero mA.	
Run-time Error: FIX (Function Index) 0	The F & P logo appears on the display.	The control program stops and the inputs are still measured. The processor continues to run but the control outputs will not be updated. The display memory is no longer updated. The contact and the analog outputs stay the same.	The FIX number determines the functionality of the controller by selecting various control strategies and operations once the FIX number is entered into the database System Module Function Index data point B000.
Failure of Display or Display Circuitry	Blank or weird display.	The processor continues to run. Both control task and display task continues to operate. The contact outputs and the analog outputs are set by control.	

FMEA of F&P 53MC5000			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
Failure of Digital Input/Output	Generally not detectable.	There could be a generic failure mode of IC, where an IC fails and causes a momentary short across the +5V supply. This generic failure mode will cause a reset of the processor. The total reset time of a 53MC5000 unit is around 1.2 to 1.5 seconds. During this time period, the analog and digital outputs of the controller will go to off state (typically ground). The failures effects on the actuators are difficult to estimate and depend on individual designs.	The failures effects on the actuators are difficult to estimate and depend on individual designs.

FMEA of F&P 53MC5000			
Failure Mode	Detection of Failure Mode	Failure Effects	Comments
ASIC Failure	Failure of the DISP-controller or the DISP-memory is visible in the display.	Loss of display.	
	Failure of the core block, the processor, will cause flashing display.	The ASIC design is hierarchical and partitioned into different blocks. If the processor fails, the watchdog timer, the DISP-controller and the DISP-memory are still working properly, which will produce a flashing display.	
	Failure of the processor's display interface or the DISP-memory (1K dual-ported RAM) is detected if the heartbeat pixel does not flash.	Loss of display.	
Clock Reference	The display will freeze and the display is possibly destroyed.	All functions of the ASIC will stop. The core block (8051 processor) will fail to execute software. Both the watchdog timer and display will freeze. Analog outputs will drift because the watchdog timer has not expired.	
Analog Output Drift	Very difficult to detect.	Under certain conditions, e.g., a total ASIC failure, the output sample and hold circuits will no longer be refreshed but the watchdog timer will not act to pull the outputs to zero. Analog outputs will drift to unknown values in unknown directions.	In critical applications, analog outputs should be monitored using input channels and a bypass circuit should be used in case of the analog output failure.

## **APPENDIX C**

### **MODELING OF SOFTWARE FAILURES**

# TABLE OF CONTENTS

	<u>Page</u>
C.1	A Unique Feature of Digital Systems: Software ..... C-1
C.2	Special Characteristics of Software ..... C-1
C.2.1	Software Life Cycle (SLC) ..... C-2
C.2.2	A Software Failure and Software Fault ..... C-3
C.2.3	Software Testing: Differences with Hardware ..... C-4
C.2.4	Does Software Age? ..... C-4
C.2.5	Software Updating ..... C-5
C.2.6	“Software-centric” and “System-centric” View of Software Failure ..... C-5
C.3	Software Failure in the Context of Digital-System Failure ..... C-5
C.3.1	Interactions Between Software and Hardware of a Digital System ..... C-6
C.3.2	Software Failure in an Operating Environment ..... C-6
C.3.3	Error Forcing Context (EFC) ..... C-6
C.3.4	Internal and External Causes of Software Failure ..... C-7
C.3.5	Propagation of Software Failure ..... C-9
C.3.6	Detection of Software Failure ..... C-11
C.3.7	Recovery of Software Failure ..... C-11
C.3.8	Common-Cause Software Failures ..... C-11
C.4	Characterization of Software Failures ..... C-12
C.4.1	Software Failure Modes ..... C-12
C.4.2	Causes of Software Failure ..... C-17
C.5	A Model Representation of Software Failures ..... C-22
C.6	Basis for Developing a Probabilistic Failure Model of Software ..... C-26
C.6.1	Considerations in a Probabilistic Failure Model of Software ..... C-26
C.6.2	Software Failure Modeling for Different Digital Systems in Nuclear Power Plants ..... C-28
C.6.3	Desirable Modeling Attributes of Software Failure Models ..... C-32
C.6.4	Review of Available Software Models in Terms of Desirable Attributes .... C-32
C.6.4.1	Review of Software Modeling Methods ..... C-33
C.6.4.2	Review of Software Failure Quantification Methods ..... C-35
C.6.4.3	Review of Methods Currently Used in PRAs ..... C-37
C.6.5	Summary of the Basis for Developing a Software Failure Model ..... C-39
C.7	Review and Analyses of Software Failures ..... C-40
C.7.1	US Nuclear Power Plant (NPP) Experience ..... C-40
C.7.1.1	Approach for Experience Data Collection ..... C-40



C.7.1.2	Summary of Results and Insights .....	C-42
C.7.2	Foreign Nuclear and Non-Nuclear Industry Experience .....	C-44
C.7.2.1	Approach .....	C-44
C.7.2.2	Summary of Results and Insights .....	C-45
C.7.3	Insights on Modeling Software Failures in Digital Systems for NPPs .....	C-47
C.8	Software Failures: Detailed Data .....	C-49
C.8.1	Software Failures in U. S. Commercial Nuclear Power Plants (NPPs) .....	C-49
C.8.2	Software Failures in Foreign Nuclear and Non-Nuclear Industries .....	C-58
C.8.3	Detailed Analyses of Selected Software Failure Events .....	C-58
C.8.3.1	Overdose of Radiation Therapy Machine THERAC-25 [Leveson 1993, Peterson 1995] - 1985-1987 .....	C-64
C.8.3.1.1	Summary .....	C-64
C.8.3.1.2	Software Failures .....	C-64
C.8.3.1.3	Consequence .....	C-64
C.8.3.1.4	Likelihood of Error Forcing Context .....	C-64
C.8.3.1.5	Failure Categorization .....	C-65
C.8.3.1.6	Dependent Failure and CCF .....	C-65
C.8.3.1.7	Discussion .....	C-66
C.8.3.2	London Ambulance Dispatch System [South 1993, Finkelstein 1996] - October 1992 .....	C-66
C.8.3.2.1	Summary .....	C-66
C.8.3.2.2	Software Failures .....	C-66
C.8.3.2.3	Consequence .....	C-67
C.8.3.2.4	Likelihood of Error Forcing Context .....	C-67
C.8.3.2.5	Failure Categorization .....	C-67
C.8.3.2.6	Dependent Failure and CCF .....	C-68
C.8.3.2.7	Discussion .....	C-68
C.8.3.3	China Airline Flight B1816 Crash at Nagoya [Ladkin, CSE] - 4/26/1994 .....	C-68
C.8.3.3.1	Summary .....	C-68
C.8.3.3.2	Software Failures .....	C-68
C.8.3.3.3	Consequences .....	C-69
C.8.3.3.4	Likelihood of Error Forcing Context .....	C-69
C.8.3.3.5	Failure Categorization .....	C-69
C.8.3.3.6	Dependent Failure or CCF .....	C-70
C.8.3.3.7	Discussion .....	C-70
C.8.3.4	Turkey Point Diesel Generator Sequencer - November 4, 1994 .....	C-70
C.8.3.4.1	Summary .....	C-70
C.8.3.4.2	Software Failures .....	C-71
C.8.3.4.3	Consequence .....	C-71
C.8.3.4.4	Likelihood of Error Forcing Context .....	C-72
C.8.3.4.5	Failure Categorization .....	C-74
C.8.3.4.6	Dependent Failure and CCF .....	C-75

C.8.3.4.7 Discussion .....	C-75
C.8.3.5 Common Cause Failure of Voltage Regulating Transformers and Vital AC Buses at Pilgrim - 4/1/1997 .....	C-75
C.8.3.5.1 Summary .....	C-75
C.8.3.5.2 Software Failures .....	C-76
C.8.3.5.3 Consequence .....	C-76
C.8.3.5.4 Likelihood of Error Forcing Context .....	C-78
C.8.3.5.5 Failure Categorization .....	C-78
C.8.3.5.6 Dependent Failure and CCF .....	C-79
C.8.3.5.7 Discussion .....	C-79
C.8.3.6 Core Protection Calculators Inoperable at Palo Verde 2 .....	C-79
C.8.3.6.1 Summary .....	C-79
C.8.3.6.2 Software Failures .....	C-80
C.8.3.6.3 Consequence .....	C-80
C.8.3.6.4 Likelihood of Error Forcing Context .....	C-81
C.8.3.6.5 Failure Categorization .....	C-81
C.8.3.6.6 Dependent Failure and CCF .....	C-82
C.8.3.6.7 Discussion .....	C-82
C.8.3.7 Slammer Virus in Davis-Besse Nuclear Power Plant [Schulin, Poulsen] - 1/25/2003 .....	C-82
C.8.3.7.1 Summary .....	C-82
C.8.3.7.2 Software Failures .....	C-83
C.8.3.7.3 Consequences .....	C-83
C.8.3.7.4 Likelihood of Error Forcing Context .....	C-83
C.8.3.7.5 Failure Categorization .....	C-83
C.8.3.7.6 Dependent Failure or CCF .....	C-84
C.8.3.7.7 Discussion .....	C-84
C.8.3.8 Natural Gas Pipeline Explosion in Soviet Union [Reed 2004, Detroit 2004, Loney 2004] - Summer 1982 .....	C-84
C.8.3.8.1 Summary .....	C-84
C.8.3.8.2 Software Failures .....	C-85
C.8.3.8.3 Consequences .....	C-85
C.8.3.8.4 Likelihood of Error Forcing Context .....	C-85
C.8.3.8.5 Failure Categorization .....	C-85
C.8.3.8.6 Dependent Failure or CCF .....	C-86
C.8.3.8.7 Discussion .....	C-86
C.8.3.9 Maroochy Water Treatment Plant Accident [Age 2003, Red 2003, Datz 2004] - 2000 .....	C-86
C.8.3.9.1 Summary .....	C-86
C.8.3.9.2 Software Failures .....	C-86
C.8.3.9.3 Consequences .....	C-86
C.8.3.9.4 Likelihood of Error Forcing Context .....	C-87
C.8.3.9.5 Failure Categorization .....	C-87
C.8.3.9.6 Dependent Failure or CCF .....	C-87

C.8.3.9.7 Discussion .....	C-87
C.8.3.10    Blackout of North America [US 2004, Jesdanun 2004] - August 14, 2003 .....	C-88
C.8.3.10.1 Summary .....	C-88
C.8.3.10.2 Software Failures .....	C-88
C.8.3.10.3 Consequence .....	C-90
C.8.3.10.4 Likelihood of Error Forcing Context .....	C-90
C.8.3.10.5 Failure Categorization .....	C-90
C.8.3.10.6 Dependent Failure and CCF .....	C-91
C.8.3.10.7 Discussion .....	C-91
C.8.3.11    Common Cause Failure of Security Computers at San Onofre Unit 1    - 1998 .....	C-91
C.8.3.11.1 Summary .....	C-91
C.8.3.11.2 Software Failures .....	C-92
C.8.3.11.3 Consequence .....	C-92
C.8.3.11.4 Likelihood of Error Forcing Context .....	C-92
C.8.3.11.5 Failure Categorization .....	C-93
C.8.3.11.6 Dependent Failure and CCF .....	C-93
C.8.3.11.7 Discussion .....	C-94

## **C.1 A Unique Feature of Digital Systems: Software**

Nuclear power plants (NPPs) are replacing their obsolete analog instrumentation and control (I&C) systems with the more reliable digital I&Cs. Digital I&Cs differ from their analog counterparts and contain many unique features that may include diagnostics, self-correction, signal validation, synchronization, and unique communication means, e.g., buses, local area networks (LAN), and fiber optic connections. Another unique feature of digital systems is the use of software.

In this Appendix, software and software failures associated with digital equipment are discussed. Often digital systems fail due to software failures, the characteristics of which may differ, requiring different approaches for preventing them and modeling for their inclusion in the plant's risk models. Recognizing the implications of shifting to a digital technology, the U.S. Nuclear Regulatory Commission (USNRC) requested the National Research Council form a committee to study the application of digital I&C technology to commercial NPP operations [National Research Council, 1977]. Among their other recommendations, the committee recommended including the relative influence of software failure on system reliability in the probabilistic risk assessment (PRA) of systems containing digital components.

The special characteristics of software failures and their influence on the failure of the digital system are discussed. Using published information, software failures are characterized in terms of software failure modes and causes, and also common-cause failures that may be associated with software. Operating experience in the nuclear industry and the experiences of other industries are reviewed to obtain further insights, that, together with the characteristics of software failures, are used to develop a model representation of software failures. This model forms the basis for software-failure modeling. A structure for addressing software failures in digital systems for NPPs is presented, along with recommendations.

## **C.2 Special Characteristics of Software**

What is Software Failure?

Some experts opine that software is deterministic, i.e., given the same input, it will always produce the same output, and argue that it may not be meaningful to try to model software [Leveson, 1991; Singpurwalla, 1995]. Some even suggest that software does not fail because it invariably does what it is programmed to do. However, it is undeniable that software does fail and has been the cause of many accidents. Also, the potential variability of the input to a software and the number of paths of execution within the software often is so large that it is impossible to exhaustively test the software. Software design faults are an important cause of software failures.

A software failure can be defined in terms of its functions and/or implied functions. A narrowly defined function of software may lead to the conclusion that the software never fails because it always does the narrowly defined function. It can be considered that any deviation from the expected behavior, e.g., a violation of one of the functions, is a failure. In a digital system consisting of hardware and software, software failures manifest themselves via the behavior of hardware.

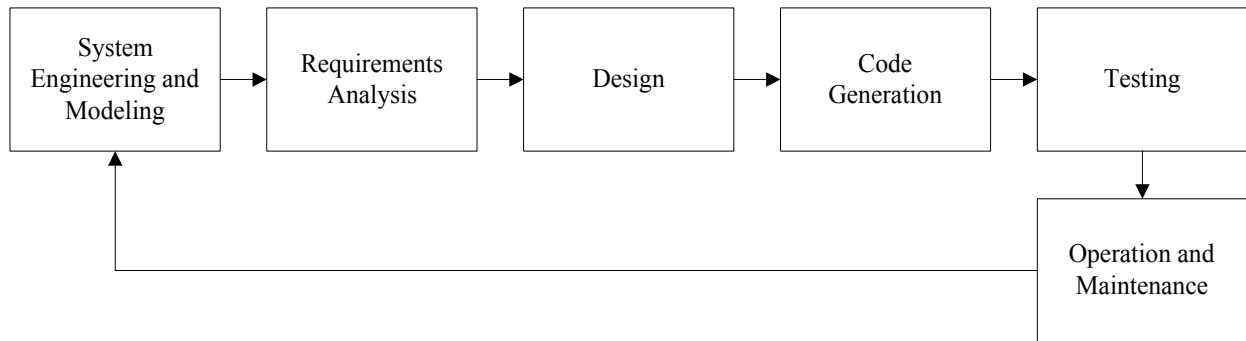
Various issues related to software failures and their different causes, characteristics, and types are discussed in the remainder of this Appendix, and are summarized in Table C-1.

**Table C-1 Characteristics of Software Failure**

<b>Software Development and Operation</b>	<b>Software Failure Concerns</b>
Software development errors	Errors may be introduced at different stages of software development. Efforts are made to eliminate them through testing. However, eliminating all errors for all feasible conditions is essentially impossible.
Software testing	Software testing identifies and corrects any errors. The process itself may introduce errors. The potential variability of the input to a software and the number of paths of execution within software often is so large that exhaustive testing is prohibitive.
Software upgrading	Software needs upgrading considering changes in the environment in which it operates. During the upgrading process, errors may be introduced resulting in software failure.
Software aging	Software does not age in the conventional sense. Certain problems, e.g., accumulation of errors over time, a memory overload, may develop over time.
Software in operational environment	The operational environment in which the software is performing may create conditions for which its design and development may not have been sufficient, resulting in its failure.

### **C.2.1 Software Life Cycle (SLC)**

Software is developed in several stages that transform it from a concept into a code that is executed by a computer processor. This development, usually called a software life cycle (SLC), is generally characterized in terms of six main stages: system engineering and modeling, software requirements analysis, software analysis and design, code generation, testing, and, operation and maintenance. Figure C.1 pictorially represents the SLC. The life cycle and each of its stages also are discussed in Section C.6 and as part of the discussion of the causes of software failure in Section C.4.



**Figure C.1 Stages in a Software Life Cycle (SLC)**

At each stage of development, errors may be introduced into the software. Thus, the requirements analysis may be incomplete, such that a requirement of the software is omitted. The earlier in the SLC an error is introduced into the software, the more severe and costly its impact is likely to be because the error is expanded in subsequent stages of development. For example, if an error is introduced during the stage of “Requirements Analysis,” the following stages will implement the error, and most likely testing will not reveal it; fixing it would require revising the entire activities of the SLC.

During “Testing,” attempts are made to discover and fix errors. However, in practice, it is difficult to discover all of them. Accordingly, some undiscovered ones may, and in many cases do, remain in the software when it becomes operational. This is particularly true for large software that can contain tens or hundreds of thousands of source code, and whose logic is complex.

Testing each possible path of execution would require large resources of time and money, so, in practice, it may be prohibitive. In addition, testing detects errors by observing that the results from certain inputs are inconsistent with the requirements of the software; if the requirements are not correct, testing will not uncover the associated errors.

### **C.2.2 A Software Failure and Software Fault**

Developing software according to high-quality standards in each stage of the SLC is expected to produce software that operates correctly during most of its operation, i.e., according to its specifications. Accordingly, in general, an undiscovered error is in a “dormant” or inactive state since the software started operating. A dormant error becomes “active” when it is triggered by a specific set of conditions. Thus, software fails when a dormant error is activated. In other words, software failure occurs from the combination of a dormant error and the specific set of conditions that trigger it. To simplify the discussion in this report, a dormant or inactive error in the software is called a “fault.”

The following are three important characteristics of a software fault :

1. It is specific to each design. Since software is developed by different teams using somewhat different approaches to the SLC, it will have its own design-specific errors. Hence, the faults may be triggered by different sets of conditions.
2. It is unknown. By the nature of a fault, it is undiscovered or hidden in the software until it is triggered.
3. The impact of a triggered fault is difficult to predict. When a fault is triggered, the software may behave in undesirable ways, having unwanted impacts on the components of the associated system. Since the fault is unknown, it is difficult to predict the effects of its failure on the software and these components.

### **C.2.3 Software Testing: Differences with Hardware**

In testing hardware, e.g., starting of a pump, *identical* tests are performed, and the results are used to estimate the failure probability. In software testing, *different* samples from the input domain of the software are used as the input. Repeating the same input in software tests does not provide any additional information. Typically, faults are fixed when they are identified. Therefore, the faults that were removed no longer contribute to software failure; rather the undiscovered faults cause concern. On the other hand, errors made in removing known faults may introduce new ones into the software. It is commonly understood that apart from simple software, it is not possible to test software exhaustively, nor can it be proven to be free of faults. Dunham and Finelli [1990] introduced the concept of error crystals in the input space representing the inputs in the input space, and which, if realized, would lead to a software failure. The removal and addition of software faults correspond to removal and addition of error crystals in the input space.

### **C.2.4 Does Software Age?**

Unlike hardware, which may fail due to physical degradation of the components themselves, software does not age. Either a software fault exists at the beginning of life, or it does not. It does not appear at some point in time except when revision to the software introduces faults. In some cases, over time, it may become impossible to recover from the accumulation of errors, e.g., the Patriot missile-defense system failed to track Scud missiles during the Iraq war [GAO, 1992], and resources, such as memory space, may become insufficient. In other words, with age and usage of the software, certain failures may occur which may not be possible at the beginning of the life cycle. If software is not updated, it may become incompatible with its operating environment when the systems with which it interfaces have changed. Arguably, these types of conditions represent aging of the software. Hence, if the software is not appropriately updated, failures will occur because of its incompatibility with the operating environment. These types of failures can be called age-related software failures.

### **C.2.5 Software Updating**

To remain compatible with operating environment and to satisfy the changing requirements for its operating system, software requires updating. This may involve minor changes to major ones. In all such updates, care must be taken to assure that additional errors are not introduced during the process. Software updating essentially involves all the stages in the SLC, and so the different types of failures that may be introduced in each stages also apply for software updates. A software that may have very low likelihood of failure at its first operation may have much higher likelihood of failure because of updating the software. Conversely, updating may correct some undetected failures in the software, making it more reliable.

### **C.2.6 “Software-centric” and “System-centric” View of Software Failure**

The characteristics of software failures, discussed above, focus on the failure of the software. However, for the digital systems in NPPs, software is a part of the digital system and in the broader context, a part of the NPP. The intent is to account for all types of software failures that may occur in the plant due to the introduction of these systems, and also for successful execution of the software within the plant’s systems.

Apostolakis [ACRS, 2005] noted that according to the literature on digital software [National Research Council, 1997], [Leveson, 1995], and [Garrett and Apostolakis, 1999], there are two main interpretations of the concept of software failure. The “software-centric” interpretation views failure as a property of the software itself. In other words, the software is considered in isolation, and not in the context of the system or the plant in which it operates. In contrast, the “system-centric” view proposes that the concept of software failure is meaningful only when considering the software within a system. This approach is very similar to the modeling of human performance; an unsafe human act is considered meaningful only in the system context within which it occurs, an observation that led the Office of Nuclear Regulatory Research (RES) of USNRC to develop the concept of “error-forcing context” (EFC) [Cooper et al., 1996]. Consistent with Apostolakis’ recommendation [ACRS, 2005], software-induced failures were analyzed (see Section C.5) and a “system-centric” view of software failures adopted that is considered applicable for capturing the different types of software failures observed. While the “software-centric” and “system-centric” views are compatible, the latter is more encompassing; it includes all “software-centric” failures and specifically accounts for the system in which the software functions. In the next section, software failures are discussed taking into consideration the digital system and their functions in an NPP.

## **C.3 Software Failure in the Context of Digital-System Failure**

It is a common understanding that software failures often reflect unanticipated inputs that the software cannot correctly respond to [Leveson, 1991; Lyu, 1994]. The variability of the input depends on the operating environment, including the systems the digital system interacts with, and the associated physical processes they operate in. The “system-centric” approach recommended by the Advisory Committee on Reactor Safeguards (ACRS) [ACRS, 2005] and supported by the analysis of software failures, as presented in Section C.5 of this Appendix, essentially asks that the



software is considered a part of the overall system in defining software failures and in modeling of these failures.

### **C.3.1 Interactions Between Software and Hardware of a Digital System**

Digital-system features are implemented in terms of both hardware and software. In general, a software failure will lead to failure of the device that it controls. In some cases, the design of the system may be such that the software can detect some hardware failures and prevent overall failure of the system. Some hardware failures may be detected by software in such a way that the failure is masked, e.g., if a sensor failure is detected, then its signal will be ignored and a second sensor's signal is used instead. However, the failure of the software to detect the sensor failure that triggers the hardware's failure because the redundant sensor's signal does not get used is a software failure of concern. Some unique features are implemented at a low level, e.g., cyclic redundancy checks [Siewiorek, 1992], communication protocol, and synchronization. A single "stuck-at-one" fault on a data line of the Traffic Collision Avoidance System of a Korean Air Cargo flight contributed to a near-miss collision with British Air flight 027 on June 28, 1999. Capturing failures relating to low-level design features may not always be feasible; even if the failures are identified at the low level, modeling of the digital system will be required at the same detailed low level.

### **C.3.2 Software Failure in an Operating Environment**

Once the software is operational, it is embedded in some system, and interacts with some environment, such as the components of the system and is impacted by the operators' actions. For example, suppose that the flow control of the Main Feedwater system (MFW) (of a Pressurized Water Reactor (PWR)) uses software. The software may control one or more components; in this example, it controls the MFW's flow control valves. In general, a software failure is propagated directly to the device(s) controlled by the software (here, the flow control valves of the MFW) that, in turn, may cause degradation or failure of the associated system.

In general, software interacts with its environment at four levels, depending on the detail: the software itself, the device(s) that are controlled by the software (e.g., the flow control valves of the MFW), the system wherein the software and the device(s) are embedded (e.g., the MFW), and the complex engineered system, such as the entire nuclear plant. Depending on the overall context of the plant and the tolerance to failures of the design of the software, device(s), and system, the failure may propagate to the overall plant.

### **C.3.3 Error Forcing Context (EFC)**

At any particular time, the software, the device(s) controlled by it, the system where the software is embedded, and the nuclear plant, are in a certain state. In general, the state of the plant provides an overall context for the operation of them. For example, the input to the software will depend on the state of the plant. Accordingly, software faults are triggered by the input to the software, which is provided by the context (state) of the plant. The context of the plant that the software "sees" is particularly important because it may trigger a fault, thus causing a failure. The context causing a

software failure was called the “EFC” by Garrett and Apostolakis [1999] because, as its name implies, it “forces” or activates the error. Accordingly, the set of conditions that trigger a fault is comprised by the EFC. This approach is very similar to the modeling of human performance; an unsafe human act is considered meaningful only in the system context within which it occurs, an observation that led the Office of Nuclear Regulatory Research (RES) to develop the concept of EFC [Cooper et al., 1996].

Figure C.2 illustrates the concept of EFC in software failures and their discovery. EFC can develop at anytime, starting from when the software becomes operational. Normal operation may be the EFC for a software that may not be discovered until some time has elapsed. A review of operational events related to software failures, in Section C.5 of this report, also showed that in several events the failure occurred practically as soon as the software became operational. In many situations, an EFC occurs sometime during the software’s operation and may remain undetected until a specific use is called for. Two representative cases of an EFC triggering a software failure are shown in the figure. Figure C.2(a) depicts the situation where an EFC occurs sometime during the operation of the software and remains dormant for an unknown period. In Figure C.2(b), normal operation is the EFC activating the fault, but its presence is discovered at a later time through a specific usage.

### **C.3.4 Internal and External Causes of Software Failure**

In addition to the failures related to the faults introduced in the software during its SLC, software may also fail due to external causes or factors, as shown by Figure C.3. Based on the review, four main types are identified: human error, failure of support systems, breaches in cyber security, and compromised environment. They are defined as follows:

1. Human error: A person may use the software inappropriately, such as using it in an operational environment for which it was not designed, or may inadvertently enter incorrect data into it. In general, these are errors by human operators that cause the software to malfunction.
  
2. Support system failures: In general, the software that performs some function in a complex system, such as controlling some device(s), requires several supporting systems. They include other software (e.g., an operating system), computer hardware (e.g., a hard drive), electric power, and possibly a system, such as a Heating, Ventilation and Air Conditioning (HVAC) system, which regulates some variables, e.g., temperature of the room(s) where the computer system is located.

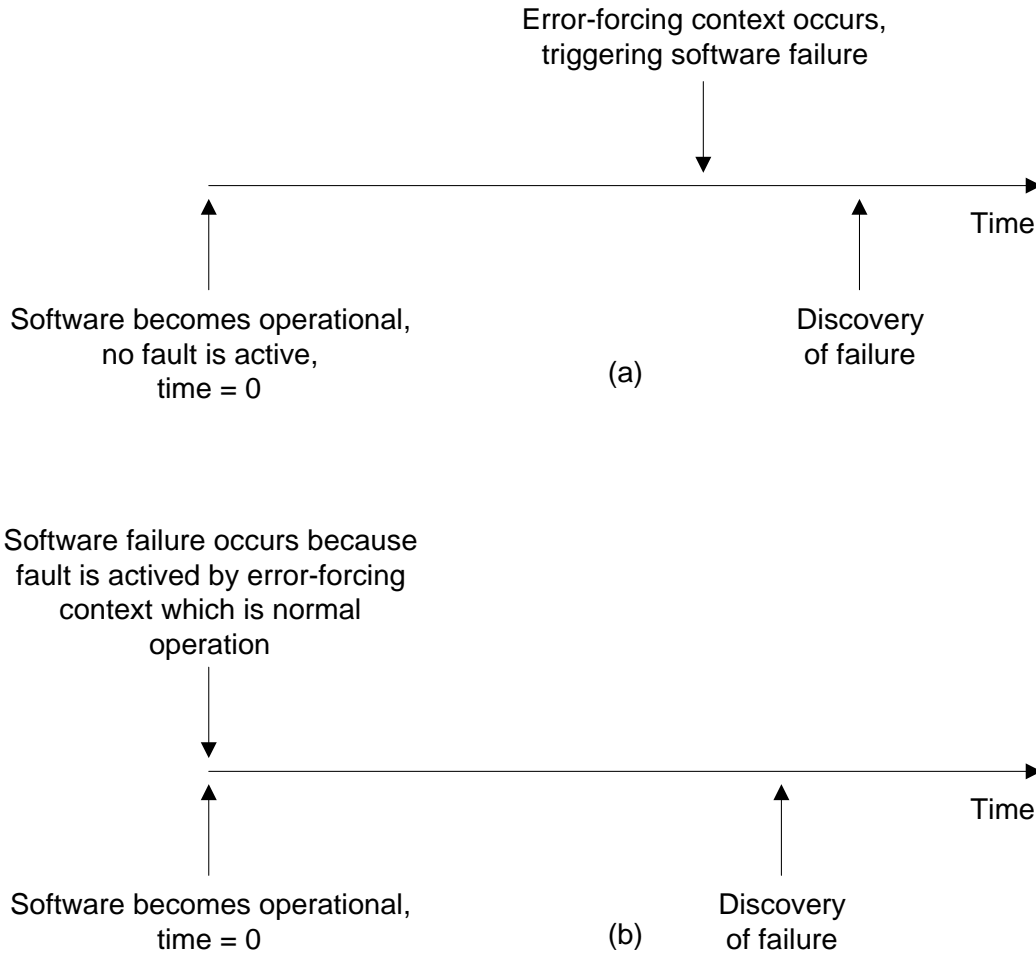
These support systems also may include an input/output interface between software and hardware. A failure in either interface may cause the software to fail.

3. Breaches in cyber security:

If the computer where the software is embedded is connected to a network, the operation of the software may be jeopardized by cybernetic threats, such as viruses and hacking activities transmitted through the network.

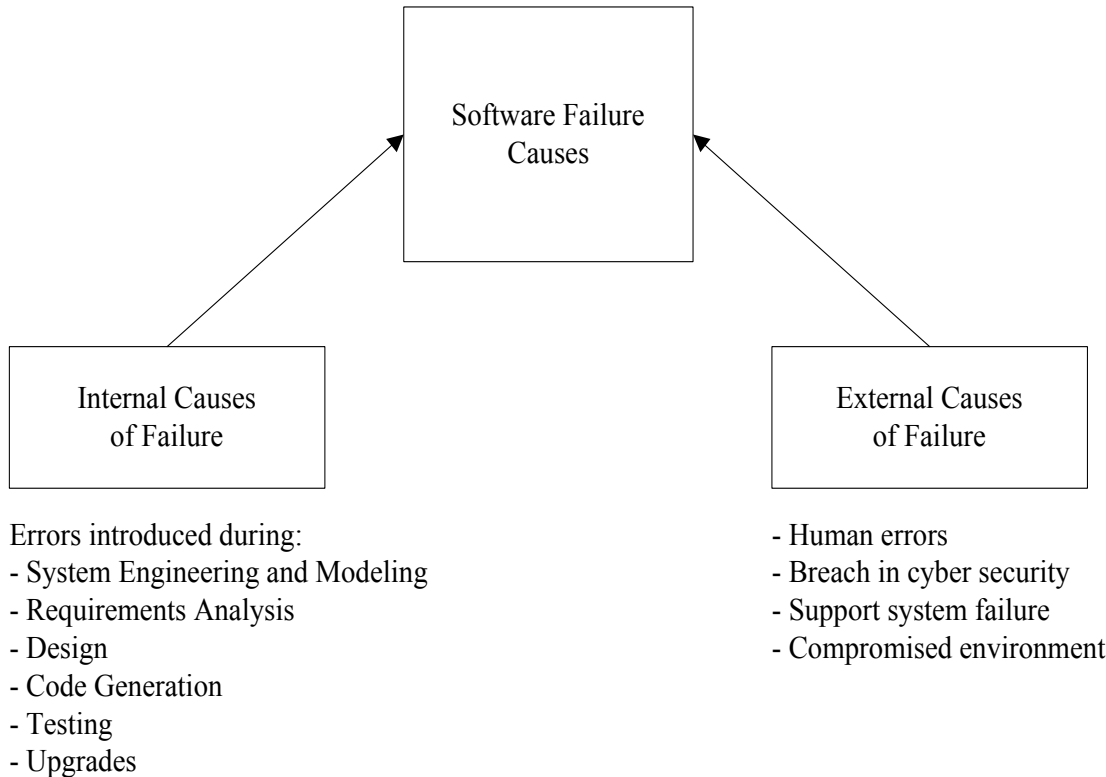
4. Compromised environment.

Events, such as fire, flooding, and lightning, also can jeopardize the operation of the computer wherein the software is embedded.



**Figure C.2 Triggering of Software Fault by Error-forcing Context**

Internal and external causes of software failures are discussed in detail in the next section.



**Figure C.3 Software Failure Causes**

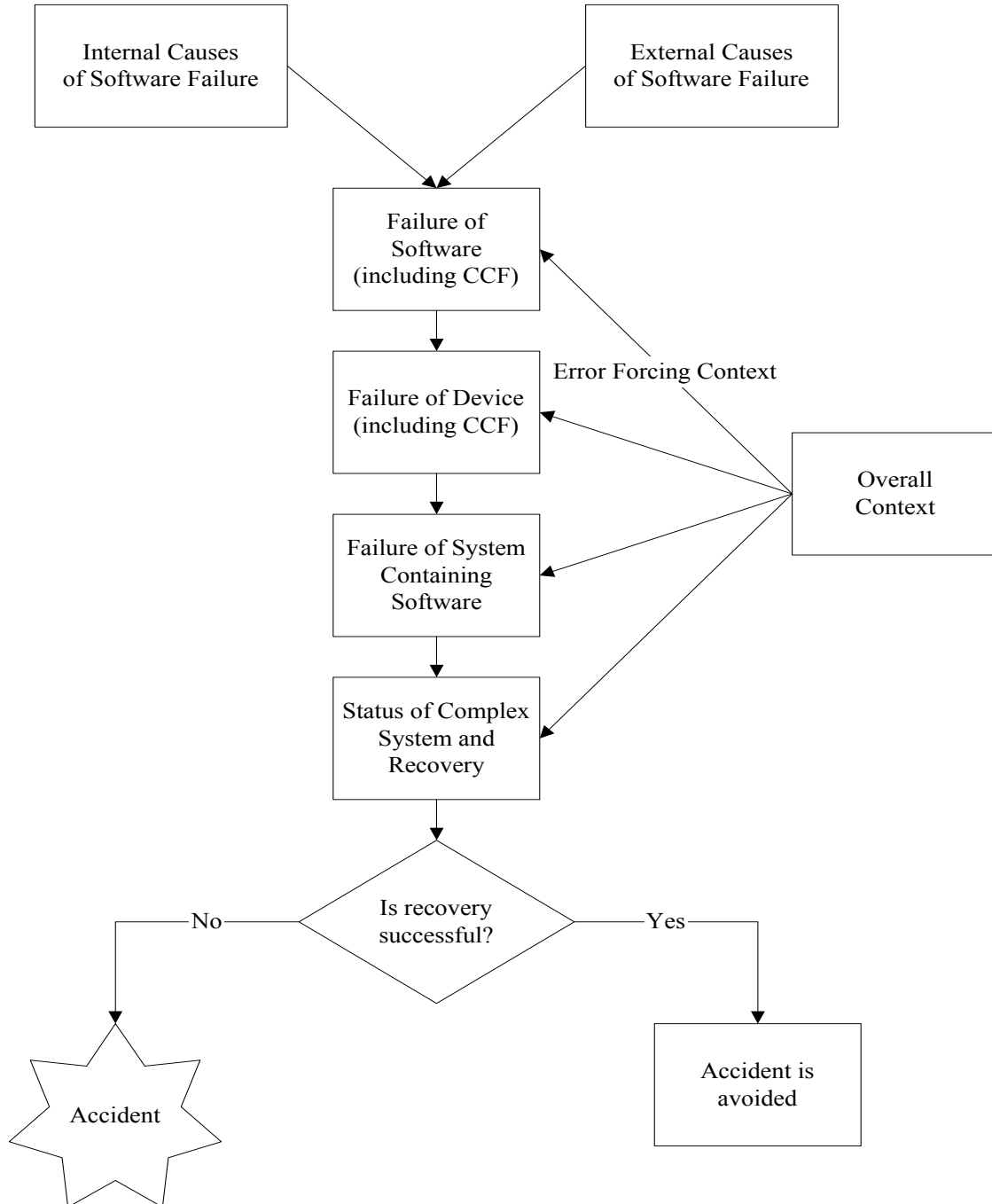
### C.3.5 Propagation of Software Failure

The propagation of a software failure to the three higher levels (device(s), the system where the software is embedded, and the nuclear plant) also depends on the plant’s overall context. For example, at the time of the software failure, other components, trains, or systems may be unavailable due to maintenance or testing, or standby components may fail to operate on demand. Hence, the software failure may combine with the unavailability of other components, trains, or systems, thus propagating throughout the plant.

In some situations, redundant components of a system or even components beyond a system may use the same or similar software that may be vulnerable to similar failures. A software failure may imply failure of similar software associated with multiple components within a circumscribed time. This is common-cause software failure (CCSF) and is discussed further below. If this CCSF occurs, it may cause all the devices controlled by similar software to fail within short time of each other and may result in the entire system’s failure.

Figure C.4 represents software failure and its propagation in a complex system. Failure of software, including CCSF, may result from both internal and external causes. As discussed, the overall context of operation of the software, the devices, and the system generates the EFC (arrow labeled “Error

Forcing Context” from the square “overall context”) that triggers a fault(s) in the software (arrow from the stage “Operation and Maintenance” of the SLC). The potential propagation of this failure is shown following the arrows downward. Thus, the square “Failure of Software” leads to the square “Failure of Device,” which leads to failure of the system containing the software.



**Figure C.4 Propagation of Software Failure**

### **C.3.6 Detection of Software Failure**

Software that was developed and operated according to a high-quality SLC is expected to initially operate without failure; any remaining faults are not active. When the software fails, it may not be evident to the plant's staff, and its detection depends on such factors as whether the failure is automatically annunciated, and whether the failure is significant to the plant's operation. In defining the EFC for a software failure, it was noted that the failure may not be detected until some time has elapsed following the triggering event caused by the EFC that resulted in the failure. If the failure is automatically annunciated, it is promptly detected and corrective measures may be taken. In some situations, a failure is significant enough to entail changes in the plant's parameters that are monitored by the staff, which facilitates detection of the failure. In other cases, the failure remains hidden and is detected only when component or system failure becomes evident.

### **C.3.7 Recovery of Software Failure**

After a software failure occurs, it may be possible to halt or mitigate its propagation and recover automatically or manually. In principle, such recovery could be implemented at any of the four levels previously discussed, i.e., at the software, device, system, or complex system (e.g., nuclear plant) level. For example, a hypothetical software failure may propagate to the system level, but recovery actions at this level can stop further propagation, so the failure does not negatively affect the overall nuclear plant. For simplicity, Figure C.4 depicts recovery only for the highest level, i.e., the complex-system level. As shown, the combination of a software failure that propagated to the complex-system level with the overall context may generate a hazardous condition that, if not recovered, becomes an accident.

### **C.3.8 Common-Cause Software Failures**

Diversity of software is difficult to demonstrate. Knight and Leveson [1986] experimented using 27 versions of a program and showed that the independence assumption is rejected with 99% confidence. The dependence came from programmers making equivalent logic errors.

Often, a digital system consists of more than one channel with identical hardware and software. Common-cause failure (CCF) of both must have to be considered. The defect found in the sequencer software logic of the Turkey Point NPP [FPL, 1994] could affect all four sequencers at the plant, and is an example of CCSF.

USNRC regulation of digital systems requires demonstration of defense-in-depth and diversity (D3) for digital systems. A D3 analysis is conducted for both hardware and software; CCFs are postulated one at a time, and diversity has to be demonstrated for each of the accidents in the final safety analysis report (FSAR) [Preckshot, 1994]. When software CCF is postulated for design-basis accidents, some accident-initiating events may become more severe than that assumed in Chapter 15 of the FSAR, e.g., the sticking open of all relief valves instead of a single relief one. The feedwater control system of ABWR [Yih, 2004] exemplifies such difficulties.

In many cases, identical software is used for two redundant components in a system. For CCSF, a complete dependence is assumed since a defect in one of the software is likely to be replicated in the software for the redundant component. In that respect, a CCSF has a severe implication compared to a common-cause hardware failure where a complete dependence is unlikely.

## **C.4 Characterization of Software Failures**

The literature on software failure modes and effects analysis (FMEA) does not yet contain a uniform definition of software failure modes, failure causes, and failure effects. A review of papers on software FMEA in the aerospace-, automobile-, defense-, and nuclear-industries, along with a review of failure experience in the medical- and nuclear-industries revealed the use of different failure classifications and confusing use of terminologies.

In analyzing digital systems for NPPs, a consistent definition of failure modes, causes, and effects is desirable since the definitions must correspond with hardware failures where a clear definition exists. For modeling digital systems and software failures in the PRA for NPPs, definitions that are comparable to hardware failures are desirable so that the models developed in this area can be integrated smoothly with the current models in the PRA.

The software system and element failure modes presented here were primarily obtained from the definitions in the literature. However, a significant improvement was made by avoiding overlaps and confusing terminology, and clearly differentiating the software- system failure modes and software-element failure modes. The intent is to clearly and consistently define of failure modes that can be used in defining software failures for digital systems in the nuclear industry.

### **C.4.1 Software Failure Modes**

In general, a failure mode represents a way a failure occurs. Software failure modes (SFMs) are difficult to define because they depend on the level of detail at which the software failure is being evaluated, and on the software's specific application. For example, failure of communication due to loss of synchronization is a lower-level software failure mode. The loss of synchronization of communication processes associated with several devices is a failure mode at a device level which may develop into a system failure impacting the complex system, such as a NPP.

Based on reviewing the literature on software failure in different industries and of the failure experience in the nuclear industry, failure modes from the view point of the dynamic execution process of software were introduced without distractions occasioned by specific functions of the software. More important, this dynamic running process exactly reflects the observed behaviors of software, i.e., the observed failures of the dynamic process are indeed the failure modes. The software SFMs presented for generic use are inspired by the work of Ristord [2001].

Software can be considered to be composed of many elements where each of the elements performs one of the generic software functions, such as input, output, processing, communication, and resource

allocation. Thus, it can be thought of as a system (software as an entity), i.e., a “software system” which consists of “software elements” performing the generic software functions.

Software can be extremely complex and each of its elements may be conceptually considered a software system (software as an entity) that again consists of many elements that can be functionally differentiated. The process (and thus the corresponding failure modes of the so-called “software system” and “software elements”) can be repeated until a level is reached where enough information is available. From this point of view, the software can be generally considered to be of a nested hierarchical structure of “software system” and “software elements,” and the failure modes can be analyzed for both repeatedly to better understand the failure modes.

Because of this hierarchical structure, it is more appropriate to separately define the software failure modes at both the system level and element level. One of the difficulties in defining generic software failure modes is that it cannot be defined according to its intended functions because a particular software has its particular function.

The published failure mode analyses were performed either at the system (software as an entity) level or at the element level (it performs one of the software generic functions, such as input, output, processing, communication, and resource allocation, which will be illustrated later), or a level that is not clearly defined. When some failure-mode classifications considered failure modes at different levels of details, more specifically at both system level and element level, the nested hierarchical structure of the software was not considered. Thus, definitions of software failure modes are presented at the system and element levels, addressing the difficulties that exist for many of the current definitions in the literature. The definition given below is considered sufficient to address different software failures observed and the modeling needs for digital systems in NPPs.

### Software System Failure Modes

Two SFMs are defined: Malfunction of software in its execution, M-I; and, Problematic, confusing, and less informative interface, M-II. These failure modes are summarized in Table C-2.

M-I represents malfunctions of software in its execution and includes two sub-modes:

#### *M-I-1 Software stalls:*

This failure mode represents software failures that stop generating output, i.e., the software runs into an infinite loop and stops generating outputs, and deadlocks between processes;



*M-I-2 Software runs as usual but with wrong outputs:*

In this failure mode, the software continues running but generates incorrect output, i.e., it accepts incorrect inputs and sends wrong outputs.

The software failure mode *M-I-1* is relatively easy to discover since all other functions cease except maybe the one it is performing. Identifying *M-I-2* is more difficult as all appears normal. Usually, for this failure mode, the overall system cannot be saved from failure.

Each of the failure modes is further expanded into two systems-level failure modes depending on whether or not the failure is clearly indicated, e.g., via an error message. Thus, four SFMs are defined for software malfunctions during execution:

SFM-1	Halt/termination with a clear message
SFM-2	Halt/termination without a clear message
SFM-3	Runs with evidently wrong results
SFM-4	Runs with wrong results that are not evident.

*M-II* represents problematic, confusing, or less informative man-machine interface (MMI) designs.

M-II Software runs with misleading commands to the user, incomplete or incorrect display of information due to software problems, missing alarms, and non- conservative output.

In this case, the software performs its intended functions successfully but contributes to human errors, or the software fails to display the information correctly. Two SFMs are defined for this failure mode:

SFM-5	Software runs with incomplete or incorrect display of information requiring the operator to take action
SFM-6	Software provides misleading commands to the operator

The failure mode M-II, which includes SFM-5 and SFM-6, were not specifically defined in the software literature that was reviewed for this study. However, this type of problem was found in many software-failure events and is too important to exclude. Another reason to consider it is that an alternate interface design very likely will prevent failure in this mode. Considering the need for modeling digital systems in NPPs and the review of nuclear experience, this failure mode was defined to assure a complete accounting of failures associated with software.

### Software Element Failure Modes

The system-level failure modes represent a natural way of considering software failure modes at the highest level. To conduct a detailed failure analysis, software-element failure modes (EFM) were introduced based on the fact that any software package can be divided into five elements which

perform the generic functions of the software. Generally, a digital system software takes input data from the hardware. Pre-processing may be performed during data input. Subsequently, the input data will be processed, and the output data is sent out. During the execution of the software, resources, such as memory and CPU, are used and communication may occur between different software processes. Therefore, software may consist of elements that can be functionally decomposed into the following:

1. SE-1: INPUT elements,
2. SE-2: OUTPUT elements,
3. SE-3: a communication element,
4. SE-4: RESOURCE ALLOCATION elements,
5. SE-5: PROCESSING elements.

Each of the five software elements is associated with failures and their failure modes can be defined. A set of generic modes is defined that apply to all elements. A set of six software EFMs is defined; they are as follows:

Timing/order failure (EFM-1):	This failure mode category represents incorrect timing and ordering of events; for example, race conditions, execution time exceeding a time limit, incorrect timing of available data, incorrect rate of data processing, incorrect duration of data for processing, and slow response.
Interrupt induced failure (EFM-2):	This category represents interrupt-induced failures, e.g., incorrect interrupt request, service, and return.
Omission of a function or an attribute (EFM-3):	This mode represents a function or its attribute (e.g., execution time of a function) being left out although they should have been included.
Unintended function or attribute (EFM-4):	This failure category represents unintended actions or attributes that were implemented, e.g., modifying variables that should not be modified, and modifying code memory, in addition to the correctly implemented intended function or attribute.
Incorrect implementation of a function or an attribute (EFM-5)	This failure mode represents a function or an attribute of the function that is not left out but is incorrectly implemented.

## Data error (EFM-6)

This category represents errors related to data. Lutz [1996] and Li [2003] listed data errors that include incorrect amount, incorrect value, incorrect range, incorrect type, absent data, duplicate data (data overflow, data saturation), corrupted output, correct input not accepted, and wrong input accepted. Here, the data error indicates problematic data that cannot be identified or handled by properly designed software logic. For example, it is impossible to design a software logic that can identify and reject a data entry with the wrong value, but within the permitted range. On the other hand, a data entry with a value that is out of the permitted range is taken as a valid input and is not considered a data error because properly designed software logic can be easily used to reject this as input. Instead, this type of failure mode is called omission of a function.

The above failure modes are applicable to all the software elements. It should be noted that for RESOURCE ALLOCATION and COMMUNICATION elements, some unique failure modes are known; they can be classified as one of the six generic failure modes listed above.

The specific failure modes include loss of synchronism, deadlock, lockout, and interruption and priority error. Examples of specific failure modes of the elements are given below:

INPUT - failure to interact with I/O board, and excessive demand on I/O devices.

OUTPUT - failure to interact with I/O board, excessive demand on I/O devices, a faulty message, checkpoint file failure, e.g., a file that describes status of hardware checked by operating system during a computer reboot.

COMMUNICATION - failed interaction (in subroutine calls, data communications) between processes, failed synchronization, dead lock (two processes prevent each other from communicating).

RESOURCE ALLOCATION - failure to interact with CPU resources, competing for resources, priority errors, resource conflicts; internal capability exceeded, dead lock (two processes prevent each other from obtaining the resource), lockout (a process is never able to acquire the resource).

Table C-2 summarizes both the software system and software element failure modes. The SFM and EFM definitions are similar to the definitions of failure modes classification in Li [2003] and could be performed at lower level according to nested hierarchical structures if enough information is available, i.e., a module also can be quite complex and might be further divided into submodules of INPUT, OUTPUT, COMMUNICATION, RESOURCE ALLOCATION, and PROCESSING, and so on.

The concept of SFMs and EFMs naturally leads to a classification of software failure analysis based on the software system’s hierarchical structure i.e., the failure mode analysis can always be described by “software system” level failure modes and “software element” level modes repeatedly. Using the failure modes shown in Table C-2, the complexity level of software failure analysis can be controlled. Moreover, the failure-mode analysis can always be performed at any level provided that the required details are available.

**Table C-2 Software System and Software Element Failure Modes**

Software System Failure Modes (SFM)		Software Elements Failure Modes (EFM)
M-I-1	SFM-1: Halt/abnormal termination with clear message	<i>Software Elements:</i> SE-1: INPUT SE-2: OUTPUT SE-3: COMMUNICATION SE-4: RESOURCE ALLOCATION SE-5: PROCESSING  <i>Generic Failure Modes of Software Elements:</i> EFM-1: Timing/order failure, EFM-2: Interrupt-induced failure, EFM-3: Omission of a required function or attribute, EFM-4: Unintended function or attribute in addition to intended ones, EFM-5: Incorrect implementation of a function or attribute, EFM-6: Data error that cannot be identified and rejected by software logic
	SFM-2: Halt/abnormal termination without a clear message	
M-I-2	SFM-3: Runs with evidently wrong results	
	SFM-4: Runs with wrong results that are not evident	
M-II	SFM-5: Incomplete or incorrect display of information requiring operators to take action	
	SFM-6: Misleading command to the user	

#### C.4.2 Causes of Software Failure

Software can fail due to errors introduced during different stages of the SLC. They are caused by the development of the software itself, and are called the “internal causes” of failure. Simply, internal causes are errors during different stages of the software development that introduced bugs or faults into the software and failed to detect them. In addition to internal causes, many software failures are related to the environment in which the software functions. Using a “system-centric” view of the software, these software failures are considered and are called “external causes” of software failure.

They are due to human errors, cyber-security problems, such as viruses and hackers, environmental problems, such as electromagnetic interferences, and failures of hardware needed to support the software. Internal and external causes together capture the different types of software failures observed in different industries. Table C.3 summarizes and briefly describes these different causes.

Six categories (*SFC-I* to *SFC-VI*) are associated with internal causes related to the stages of the SLC; the remaining four cover the external causes. Note that problems with documentation may become a cause of failure at each stage of SLC. Table A.21 in IEEE [1995] lists the documentation problems, so they are not listed herein.

**Table C-3 Software Failure Causes**

<b>Software Failure Cause Type</b>	<b>Software Failure Cause</b>		<b>Cause Description</b>
Internal Causes	SFC-I	System Engineering and Modeling	Software developed with problematic or outdated standards or policies, and cannot integrate with the overall system
	SFC-II	Software Requirement Analysis	Incomplete or incorrect requirements for developing software
	SFC-III	Software Analysis and Design	Failure to include desired functions of software, and adoption of improper algorithms, methods, and structures of individual parts of the software
	SFC-IV	Software Code Generation	Introduction errors, commonly known as bugs, in the software product because the software was not coded as intended.
	SFC-V	Software Testing	Grossly insufficient or inappropriate testing before releasing f the software
	SFC-VI	Software Operations and Maintenance	Errors introduced when modifying the software
External Causes	SFC-VII	Breach of Cyber Security	Errors caused through breach in cyber security, e.g., hacking and introduction of viruses
	SFC-VIII	Human Error	Operating errors that cause software to malfunction, e.g., an operator mistakenly issues incorrect command after which the software fails to perform the function the operator intended it to do
	SFC-IX	Support System Failure	Failure in an input/output interface between software and hardware, and other hardware resources, such as a CPU, memory, or disk space that support computer system's normal operation

Software Failure Cause Type	Software Failure Cause		Cause Description
	SFC-X	Compromised Environment	Environmental causes may include damages from lightening strikes , improper operating temperatures, fire, and flooding.

### Internal Causes of Software Failure

Six categories of internal causes of software failure related to the stages of a SLC are defined as follows.

*SFC-I System engineering and modeling:* An example of failure caused by problems at this stage is that the software cannot be integrated into the overall system. Some typical causes are:

- SFC-I-1* Incompatibility between software and overall system
- SFC-I-2* Using problematic or outdated standards/policies

*SFC-II Software requirements analysis:* The failure causes at this stage include incomplete or incorrect requirements of software. An example is that certain functions the software should perform were not specified (and thus not coded in the software). Typical examples include:

- SFC-II-1* Conditions that might impact on a specific function are not taken into account, e.g., exception condition is not specified
- SFC-II-2* Missing functions; the desired functions are not specified in the requirements
- SFC-II-3* Incorrect specifications; the desired specification exists but is incorrect

*SFC-III Software analysis and design:* Failures at this stage include failure to include desired functions of the software, and adoption of improper algorithms, methods, and structures of individual parts of the software.

Timing interaction between data and processes is more critical for a real-time digital system. For non-real-time systems, communication failure between multiple processes might also be caused by this issue. Some of the general causes at this stage are:

- SFC-III-1* Calculation; the improper handling of boundary conditions, incorrect equations, missing calculations, an incorrect operand or operator in equations, parentheses used incorrectly, incorrect handling of abnormal conditions,

precision problems, rounding or truncation errors, sign convention errors, units' conversion errors.

*SFC-III-2* Algorithm; the selection of an improper algorithm, algorithm error induced unintended function.

*SFC-III-3* Logic; forgotten cases or steps, duplicated logic, extreme conditions neglected, misinterpretation of logic, missing condition testing, checking wrong variables, an incorrect iterating loop, configuration scheme for component interaction allows incorrect behavior, logic error induced unintended functions.

*SFC-III-4* Data handling (manipulation other than computation); incorrect data initiation, data accessed or stored incorrectly, incorrectly setting for flags or indices, incorrectly packed/unpacked data, data referenced out of bounds, incorrect scaling or units, incorrectly dimensioned data, incorrectly subscripted variable, improper data validation.

*SFC-III-5* Fault tolerance; incorrect action due to abnormal conditions or human errors, exception handling.

*SFC-III-6* Interface; the software does not properly interface with external device or other software components, such as mismatched or incorrectly called subroutines, subroutines called at the wrong location, a nonexistent subroutine called, inconsistent subroutine arguments (e.g., inconsistent types of arguments).

*SFC-III-7* Temporal faults; the processes are out of synchronism with each other or waiting for each other for further actions, inaccurate clock or timer failure, interrupts handled incorrectly.

*SFC-IV Code generation:*

The failure causes at this stage may introduce the errors, commonly known as bugs, in the software product because it was not coded as intended. Thus, it does not function as expected in certain situations even if there is no problem with previous stages of development, such as "requirement analysis" and "analysis and design." Examples of typical causes of failure in this stage include:

*SFC-IV-1* Typos: misspelled variables, incorrect variable usage, e.g., referencing the wrong data variable

*SFC-IV-2* Functions not coded although designed

*SFC-V Testing:*

The failure causes that occur at this stage might be due to grossly insufficient or inappropriate testing before releasing the software. Some of the causes are:

*SFC-V-1* Incomplete test plan and/or test procedures

*SFC-V-2* Test plan was not implemented or executed appropriately

- SFC-V-3 Regression test<sup>1</sup> was not performed on modified software
- SFC-V-4 Untested for different running environments that might be encountered
- SFC-V-5 No validation before initial release
- SFC-V-6 No validation on software changes
- SFC-V-7 Problem with the quality assurance plan problem

*SFC-VI Operation and maintenance:* The failures at this stage can be caused by modifying the software. Some of the typical causes include:

- SFC-VI-1 Improper upgrades of software because of wrong procedures,
- SFC-VI-2 Failure to upgrade related systems, including both software and hardware, such as incompatibility between the upgraded software and the existing hardware,
- SFC-VI-3 Problems with the software configuration plan, maintenance plan, and product-support plan after installation or upgrades,
- SFC-VI-4 Issue with the management of the software configuration ,
- SFC-VI-5 System administration, e.g., incompatible operating system caused software failure.

Four categories of external causes of software failures are defined here.

*SFC-VII Breaches in Cyber Security* Cyber security issues include introducing viruses and hacking.

*SFC-VIII Human error* Human causes include operating errors that cause the software to malfunction, e.g., an operator mistakenly issues an incorrect command so that software fails to perform the function the operator intended it to do.

*SFC - IX Support System Failure* Support system causes of failure may include an input/output interface between software and hardware, and other hardware resources, such as a CPU, memory, disk space etc., that support the computer system's normal operation.

*SFC-X Compromised Environment* Environmental causes may include damages from lightening strikes, improper operating temperatures, fire, and flooding.

---

<sup>1</sup> Regression testing is a type of software testing that seeks to uncover regression bugs. Regression bugs occur whenever software functionality that previously worked as desired stops working or no longer works in the same way than t previously planned. Typically, they occur as an unintended consequence of program changes. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged.



The failure of the software can be caused by incorrect input, due to either hardware failures or human errors. If the faulty input appears normal and valid (e.g., within the range of predefined input values), software or hardware logic cannot be built to identify and reject it. In this case, the cause of the software failure is support-system failure or human error.

On the other hand, if the incorrect input due to hardware failures or human errors that causes the software failure is detectable with properly designed software logic, then the cause of this failure is a missing desired function (*SFC-II-2*, the software logic should be designed to detect the abnormal input). This is a major difference between failure causes *SFC-II* and *SFC-VII*.

Note that if the wrong inputs due to human or hardware that could be captured by properly designed software logic but are not captured because of the lack of the software logic, the failure cause should be called missing function in a software requirement analysis stage.

## **C.5 A Model Representation of Software Failures**

In this section, a conceptual model of software failures is presented (Figure C.5). This conceptual model considers the stages of software development wherein failures may occur, the EFC that may develop during operation, and the propagation of these failures in a complex engineered system. The model in this figure was inspired by a review of the relevant literature (especially, Yih et al. [2005], and Garrett and Apostolakis [1999]), and the review of operational events related to software failures, discussed in the previous section. Essentially it combines the software errors that may have been created during the software's life cycle with the software triggering faults created by the EFC during operation of the software in a system. Propagation of software failure within the system onto the entire system then is considered to arrive at a complete picture of software failures.

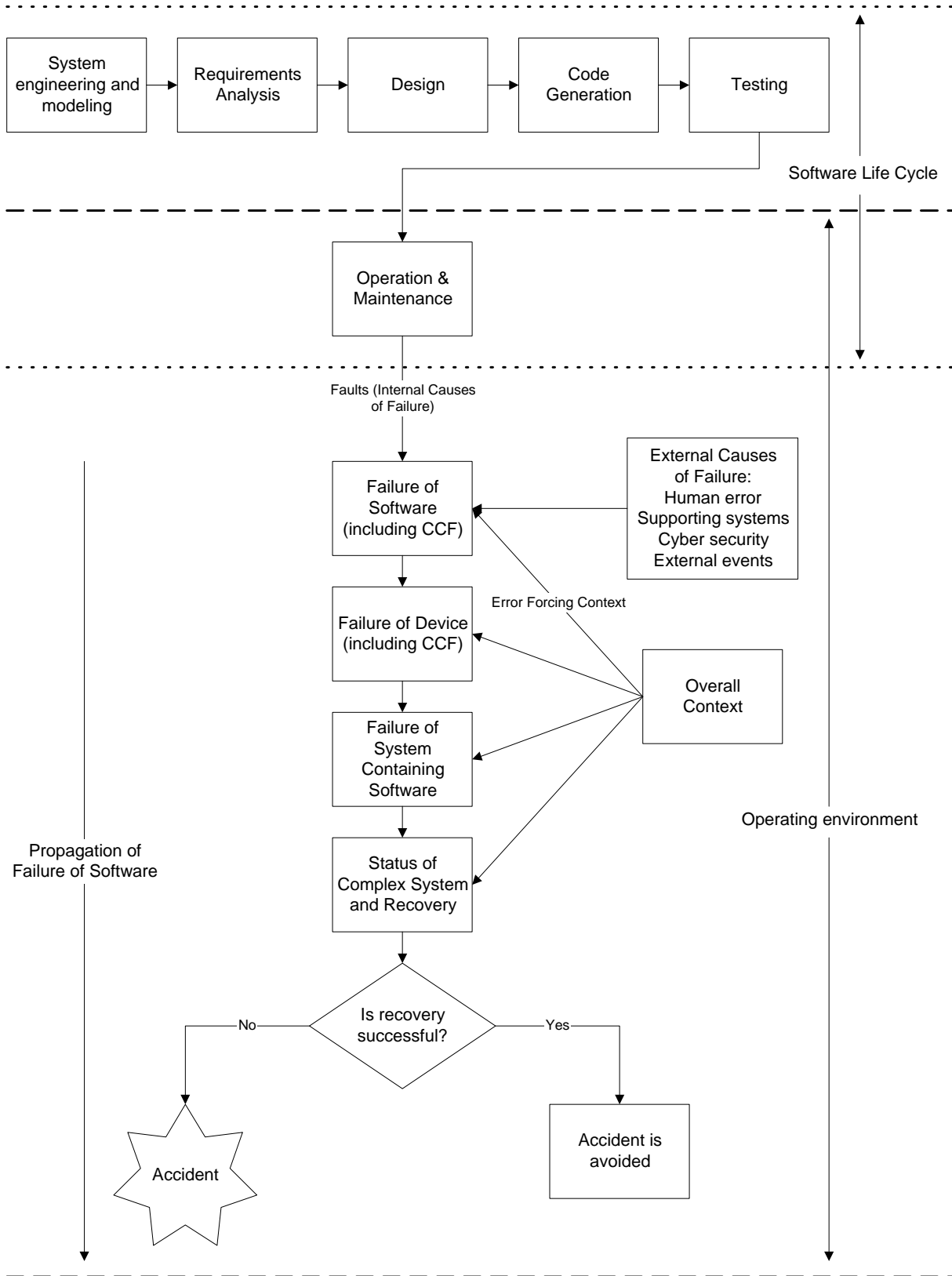
The SLC is defined in six stages that transform the software from a concept into a code that is executed by a computer processor. A software failure can originate in any of them:

1. System engineering and modeling. This is the beginning of the software development. Since software is part of a bigger system, the transformation from a concept begins by establishing requirements for all the system's elements, and then assigning some subset of these requirements to software. This system view is essential when software must interface with other elements, such as hardware, people, and other resources. The system will be engineered or designed before its development team specifies the requirements for the defined system.
2. Software requirements analysis. The analysis of requirements focuses on the software, as well as its function, behavior, performance, and interface with the rest of the system. Among the parameters considered should be software product's ability to handle unexpected or false input signals, display for human-machine interfaces, and temporal interactions. Accordingly, the development team formulates documents containing the software's specific technical requirements.

3. Software analysis and design. The analysis and design process translates those requirements into a conceptual representation of the software that can be evaluated before coding begins. The overall structure and the details, such as the logical- and algorithmic-systems, database design, and data structure design, of the software will be defined.
4. Code generation. This stage translates the software's design into a computer programming language, usually simply known as code, that a computer can understand. If the design is detailed, the coding can be accomplished without complications.
5. Testing. Some of the aforementioned stages may include testing. In this stage, testing is carried out for the overall software after integrating individual modules. The tests are meant to uncover any errors. Generally, this is achieved by observing that the results obtained from certain inputs are consistent with the requirements of the software.
6. Operation and maintenance. The software is installed in its operational environment, and operation starts. The software should be developed to accommodate changes that could happen during the post-implementation period. After the software has operated for some time, it may require modification for many reasons, for example because errors were discovered. When maintenance is required, in principle, the previous five stages must be repeated before the needed change, so that it is incorporated correctly into the software and associated system, and no additional errors are introduced.

The upper part of Figure C.5 (between the dotted lines) presents the SLC that uses the “linear sequential” model of development. As this name implies, and as depicted in the figure, the stages of the SLC are conducted in sequence. While this is an old model of SLC, and newer ones are available, it is used here as a conceptual framework for developing software, and to avoid unnecessary complications.

At each stage of development, errors may be introduced into the software. Two examples are that the requirements analysis may be incomplete, such that a requirement is omitted, and an error may be introduced during the stage of “Code Generation,” usually known as a “bug.” The earlier in the SLC an error is introduced into the software, the more severe and costly its impact is likely to be because it is expanded in subsequent stages of development. For example, if an error is introduced during the stage of “Requirements Analysis,” the following stages will implement it, and, most likely, testing will not reveal it; fixing it requires revising the entire activities of the SLC.



**Figure C.5 Model of Software Failures**

While the stage of “Testing” attempts to discover errors so they are fixed, in practice it is difficult to discover all of them. Accordingly, some undiscovered errors may, and often do remain in the software when it is installed in its operational environment and becomes operational. This is particularly true for large software that can contain tens or hundreds of thousands of source code, and whose logic is complex. Testing each possible path of execution would require tremendously large resources in terms of time and money, so in practice it may be prohibitive. Further, testing detects errors by observing that the results from certain inputs are consistent with the requirements of the software; if the requirements are incorrect, testing will not reveal some errors.

The software errors may originate during the SLC and remain embedded in the software when it is operated as part of a system. As discussed earlier, these dormant errors may be activated when an error forcing context develops, as shown here. The software failure can then propagate, depending on the design of the system in which it is used. In addition to errors developed during the SLC, external causes can also cause a software to fail, which also is included. If the failure is detected and recovered, any adverse effect is averted. Otherwise, the failure can lead to a failure of the system and, ultimately, to an accident with undesirable consequences.

The above model representation of software can be employed to identify the following modeling needs for integrating software failure in digital systems as part of a model for the overall system, e.g., the PRA for an NPP.

1. Software failures can be modeled considering the occurrence of triggering events that generate from the EFC that activates dormant errors created during the SLC.
2. Software failures from external causes may need to be modeled separately, in addition to the model developed for internal causes during the SLC. The likelihood of occurrence of external causes may need to be developed to model software failures from external causes.
3. Software is susceptible to common-cause failures. If redundant components or systems use the same equipment containing the same software, then such failures should be considered. For software, complete dependence may need to be assumed.
4. Detection of, and recovery from software failures depend on its features and the system in which it is used. Software failure detection and recovery may need to be included as part of the system model.
5. Propagation of the software failure in the overall system is expected to be modeled similarly to other failures modeled in the overall system. CCSF for multiple systems that play a role concurrently in the overall system may need consideration.

## **C.6 Basis for Developing a Probabilistic Failure Model of Software**

### **C.6.1 Considerations in a Probabilistic Failure Model of Software**

The software being considered is one that is put into operation after the development stages are completed. In other words, the software has gone through the testing stage to address identified corrections, and is operating in the overall system for which it is designed. A “system-centric” view of the software failures is employed, and the considerations that apply for probabilistic modeling of software failures are defined.

#### Operating environment, input space, and operational context

The operating environment of a digital system and its software include all aspects that either directly or indirectly affect the input to the software. In a NPP, the operating environment of a digital system practically involves the entire plant and possibly, some offsite equipment, such as offsite power, since the digital system may receive input from many systems and their support systems. For example, for the digital feedwater control system at a NPP, the operating environment would include the physical processes in the reactor coolant system and related systems, the support systems, and operator actions. The changes in the operating environment generate changes in the input to the software and define the input space. Thus, the operating environment defines the possible inputs that the software would encounter, i.e., the input space. In general, the input space is a continuum, not well defined, and the probabilistic distribution of the input over the input space, i.e., the operational context of the software, is not precisely known [Frank, 1997].

In summary, the operational environment and the probabilistic distribution of the input over the input space, i.e., the operational context of the software, should be considered for modeling software failures. This would assure that they are not narrowly defined and that all relevant ones are covered.

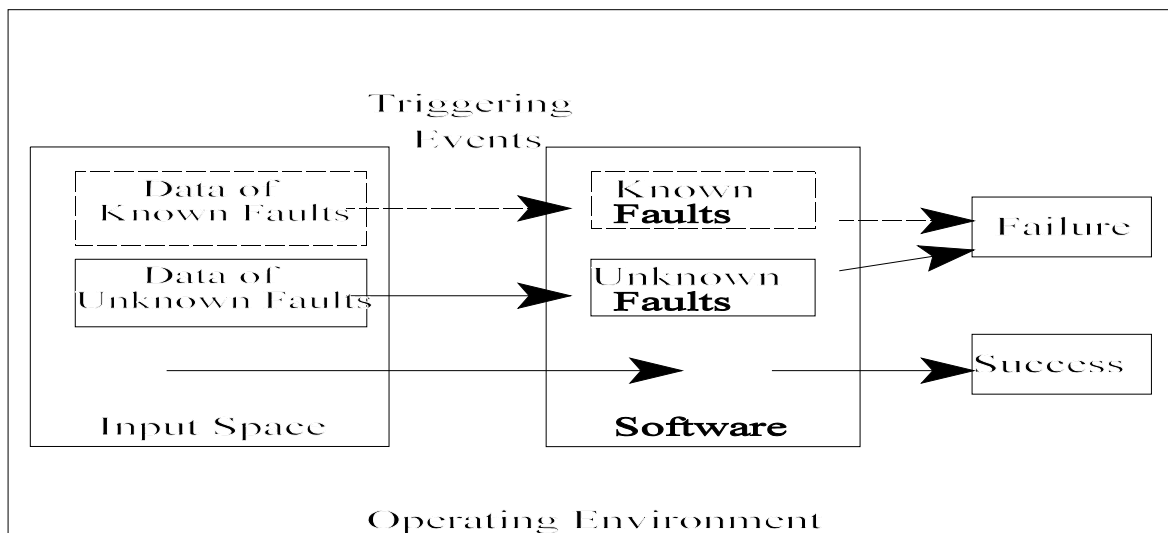
#### Faults, failure triggering events, and failure modes

The software development process is not expected to be perfect and hence, some faults may remain in the software. A fault can be activated/triggered if certain inputs in the input space take place, which can lead to a software failure. Figure C.6 is a mapping of the parts of the input space that would trigger faults, i.e., error crystals [Dunham, 1990], through the triggering events, to the failure state.

If a fault is identified and corrected, then the corresponding triggering events are no longer applicable events. If a new fault is introduced, for example when revising the software, then the new corresponding triggering events are applicable. The occurrence of the input that triggers a particular fault is random, and its likelihood is a function of the operating environment.

The likelihood of a software failure is the same as that of the triggering events, given that a fault was introduced. Depending on the modeling needs and the function of the software, the likelihood of software failure can be expressed in terms of a frequency or probability, i.e., that of the triggering event; that is, for a continuously running control system, a failure rate is applicable, while for an actuation system, a demand probability is appropriate.

A software may fail in any one of its failure modes, i.e., it may have different modes with different effects. A failure rate or failure probability considering the corresponding triggering event may apply to each failure mode. Modeling should cover the identified failure modes considering the randomness of the triggering events that apply to the failure mode.



**Figure C.6 A Framework for Probabilistic Failure of Software**

The randomness in the occurrence of the triggering event is the source of uncertainty in the occurrence of software failures. For the types of uncertainties in modeling software failure, e.g., aleatory and epistemic uncertainties are applicable, the standard interpretation of how software-failure-triggering events occur [Parry, 1996]. For example, for a control system, the aleatory uncertainty of software failure can be modeled by a Poisson process with a failure rate, and the epistemic uncertainty by the parameter uncertainty associated with the failure rate.

Assuming that the occurrence of software failure is random because the fault-triggering event is random, it is judged that modeling of software failures in terms of a failure rate or failure probability should cover the random nature of software failures. The basic premise here is that it is meaningful to model software failures using this concept. Figure C.6 depicts this concept as a framework for modeling software failures. This framework applies to the digital systems at a NPP but is generally applicable to the digital systems of other industries. Any software reliability model of a digital system is expected to be consistent with this framework.

The considerations for developing a model for software failures can be summarized as follows:

1. The operating context of the software that includes the operating environment and the distribution of the inputs that may be generated in the input space should be taken into consideration in modeling software failures.
2. The software failures are random due to the randomness of the triggering events that caused them. Hence, software failure can be modeled in terms of failure rate and/or failure probability and their associated uncertainties.

### **C.6.2 Software Failure Modeling for Different Digital Systems in Nuclear Power Plants**

For modeling purposes, the instrumentation and control systems of a NPP can be categorized into three different types:

1. Control systems,
2. Actuation systems, and
3. Indication and alarm systems that support the operators' actions.

Control systems are systems that are normally operating, e.g., the feedwater control system and the chemical and volume control system; their failures may cause a plant transient, e.g., a reactor trip, or an accident-initiating event, e.g., a Loss of Coolant Accident (LOCA). In general, failures of digital control systems may generate events that are different from, and possibly more challenging than, those considered in design-basis accidents. For example, software common-cause failure may cause redundant equipment to fail, so challenging the design's safety margin [Yih, 2004].

Actuation systems are systems that are normally on standby, e.g., the Reactor Protection System (RPS) and the Emergency Safety Features Actuation System (ESFAS), and should generate the needed actuation signals when demands arise. An actuation system may spuriously generate a signal causing a transient, or fail to do so when there is a demand.

Indications are normally available for operators to monitor the plant's status, and alarm systems alert the operators about abnormal conditions that may require their action. An indicator is normally running and may fail to provide correct information to the operator, leading to human errors of omission and commission. An alarm system is normally on standby, and would generate an alarm(s)

on demand; its failures could entail the same types of human errors as failures of an indicator, except that the failure modes are false alarms and failures to alarm on demand.

These three types of instrumentation and control systems are also characterized by different types of failure that influence the software reliability model. For a control system, software failure occurs when a set of input signals triggers a fault in it. The occurrence of the set of signals can be modeled in terms of a frequency that is a function of the operational environment of the control system.

An actuation system, during standby, may fail, generating a spurious actuation signal. Its occurrence reflects a set of input signals triggering a fault of the software, and can be modeled in terms of a frequency, similar to the failure of a control system. When there is an actual demand for the actuation system, then it may fail because the input signals of the demand trigger a fault in the software. The occurrence of the demand can be modeled in terms of a frequency, and the occurrence of the fault-triggering input signals can be modeled as a failure probability, i.e., the probability that the input signals correspond to an EFC.

The failure of an indicator to provide correct indication can be modeled in terms of the frequency of fault-triggering events, similar to a control system, except perhaps that the indicator's failure may not be recognized immediately. For an alarm system, a spurious alarm or a failure to alarm can be modeled in the same way as is spurious actuation or failure to actuate an actuation system.

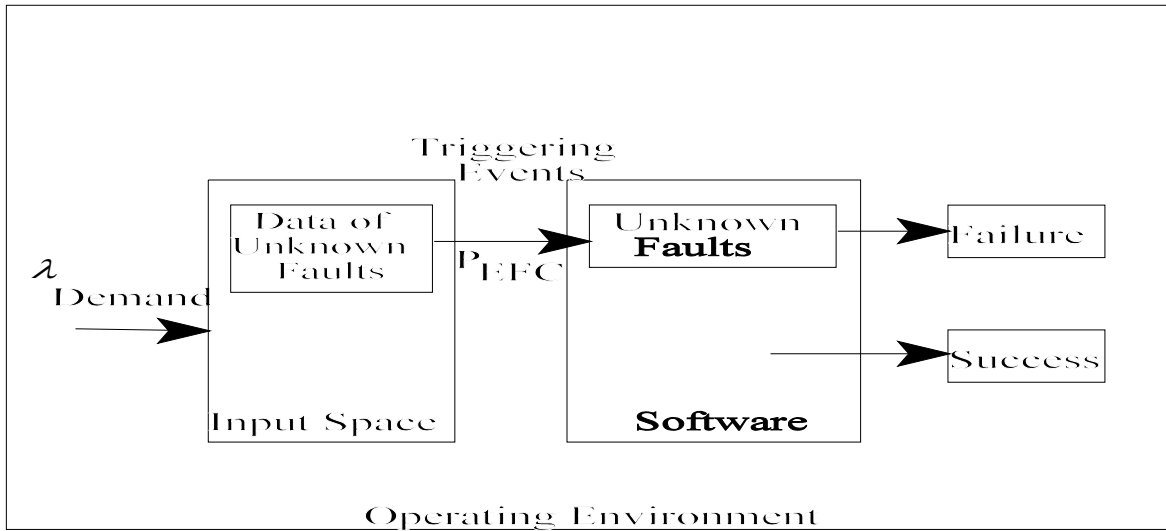
Figure C.7 illustrates two types of models needed for digital systems of a NPP in terms of failure rates and of demand failure probabilities. Table C-4 summarizes the different modeling considerations for different types of digital systems.

For a control system, the occurrence of the input that would trigger a software failure is represented by a failure rate,  $\lambda_{\text{EFC}}$ . For the failure mode of a spurious actuation signal of an actuation system, the occurrence of an EFC that triggers a spurious signal can be represented by  $\lambda_{\text{SS}}$ , and can be modeled in the same way as is failure of control systems. For failure to generate an actuation signal from an actuation system, a demand for the actuation system must be generated, represented by a frequency  $\lambda_{\text{Demand}}$ . In general, different kinds of demands may have to be considered separately. For example, a LOCA and a loss of feedwater both require the RPS to trip the reactor, but they represent different challenges/input signals to the RPS, and have different frequencies, and therefore, may necessitate modeling them separately. Given a demand on the actuation system, the input to the system may be such that it would trigger a software failure; this can be represented by a probability, i.e.,  $P_{\text{EFC}}$ . For different demands on the system, different probabilities may have to be used because the demands have different input spaces. Models for the failure modes of indicators and alarms can be formulated using the two models in Figure C.7.

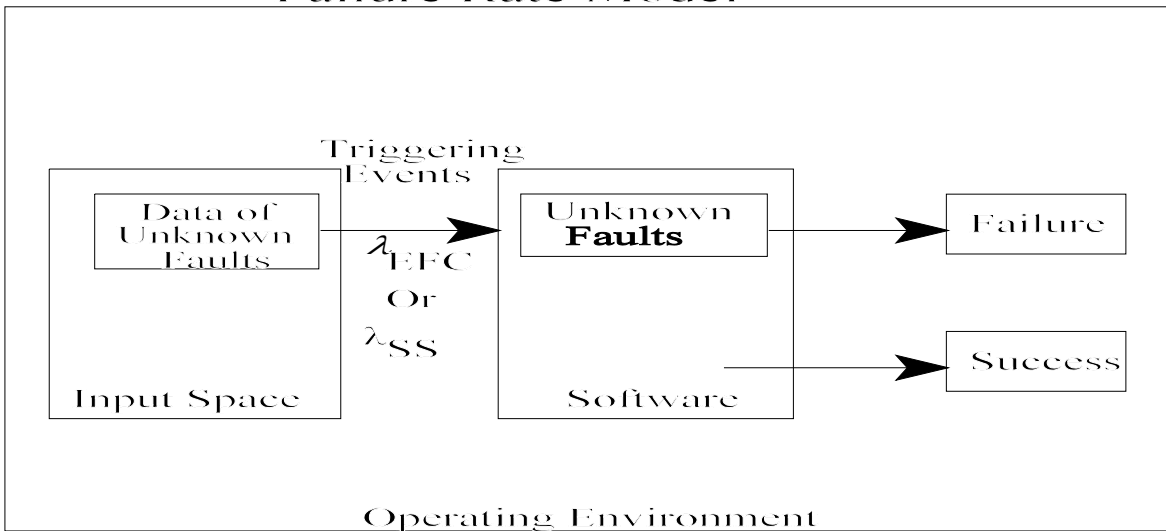
A digital system may operate in different modes due to the plant's different operational modes, e.g., full power, low power, and shutdown. Due to these differences, the digital system operates in significantly different operational environments, and probably has to be modeled separately to accommodate them. An operational mode of a digital system defines its operational environment and the associated operational context and input space.



### Demand Failure Model



### Failure Rate Model



**Figure C.7 Probabilistic Failure Models of Software**

**Table C-4 Considerations in Software Failure Modeling for Different Types of Digital Systems**

<b>Type of Digital System</b>	<b>System Characteristics</b>	<b>Failure Characteristics</b>	<b>Failure Modeling</b>
Control System	Continuously operating	A set of input signals triggers a fault in the software	Model occurrence of the input that would trigger a software failure in terms of failure rate
Actuation System	Standby - Failure results in a spurious signal	Spurious actuation signal. Occurrence of an EFC triggers a spurious actuation signal	Model occurrence of EFC that triggers a spurious actuation signal by failure rate
	Standby - demand to generate an actuation signal	Failure to generate an actuation signal on demand. Input signal triggers a fault in the software	Occurrence of the input requiring a demand can be modeled in terms of frequency  Occurrence of the fault-triggering input signal can be modeled in terms of a failure probability.  May require consideration of different types of demands separately with different failure probabilities
Indicator and Alarm Systems	Indicators - continuously operating	Failure to provide correct indication Input triggers a fault in the software	Model occurrence of the input that would trigger a software failure in terms of failure rate
	Alarms - spurious actuations	Spurious actuation signal. Occurrence of an EFC triggers a spurious actuation signal	Model occurrence of EFC that triggers a spurious actuation signal by failure rate

Type of Digital System	System Characteristics	Failure Characteristics	Failure Modeling
	Alarms - failure to alarm on demand	Failure to generate an alarm signal on demand Input triggers a fault in the software	Occurrence of the fault-triggering input signal can be modeled in terms of a failure probability.

**C.6.3 Desirable Modeling Attributes of Software Failure Models**

The above discussions of considerations for modeling software failure and different types of digital systems in NPPs and their modeling needs offer an idea of the required attributes for software models. Essentially, they are ones that will assure the model(s) selected from those available, or any newly developed models will possess the desirable features so that different types of failures can be included.

The attributes cover the ability to model the following parameters:

- Normal behavior of the system
- Software failure in terms of a failure rate or probability
- Normal behavior of hardware
- Hardware failure in terms of failure rate or probability
- Interactions between software and hardware
- Explicitly consider the contexts that affect the input to the software
- The model should demonstrate overall consistency with these desirable attributes.

**C.6.4 Review of Available Software Models in Terms of Desirable Attributes**

The available models for software failures were reviewed to assess their suitability to meet the needs in modeling digital systems for the PRA of NPPs. Different types of digital systems, their failure characteristics, and the defined modeling attributes were considered. The review focused on the scope of the applications and the ability of the models to address the attributes defined, but did not consider the correctness of the application, the completeness of the modeling, level of modeling detail, or the availability of needed data.

The methods were further divided into three groups: software reliability modeling methods, software failure quantification methods, and currently used methods in the PRA. The first group consists of methods for modeling digital systems whose application can potentially be used in a PRA, and the second group encompasses methods for quantifying software-failure rates and probabilities that can potentially be used in different models and applications. The third group comprises the methods now

used in PRAs for modeling human- and hardware-failures. The reviews of the individual methods are summarized next; Table C-5 summarizes the models and their applicability.

#### **C.6.4.1 Review of Software Modeling Methods**

The methods for modeling digital systems were examined to determine their ability in probabilistic-failure modeling of software; only two explicitly model failures of software, i.e., AP600 PRA [Westinghouse, 1996] and the NASA software conditional risk model [NASA, 2002]. Neither gives a method to quantify software failure rates and probabilities. The dynamic Flowgraph Methodology (DFM) [Garrett, 1999] in principle captures software failures by explicitly modeling its normal behavior. All other methods either model the normal behavior of software, e.g., formal methods [Clarke, 1996] or simply assume that it does not fail, e.g., the Markov model of Tricon [Swanner, 2000]. Those methods that model its normal behavior, e.g., Petri nets [Sivertsen, 2004] and formal methods [Clarke, 1996] potentially can be used to identify software faults and improve the quality of the software. Two “dynamic” methods, i.e., dynamic Markov model [Aldemir, 2006b] and DFM [Aldemir, 2006b] explicitly model the physical processes and related plant systems and, advantageously, are explicitly “system centric.” More detailed discussion of each method is given below.

Fault tree analysis of AP600 [Westinghouse, 1996]- Its modeling of software for actuation systems, e.g., RPS and ESFAS, is in the form of basic events representing software common-cause failures in a fault tree. The failure probabilities appear to be arbitrarily assigned and are described as reliability goals. The CCF basic events effectively appear at the top of the systems’ fault trees of the systems. Using CCF basic events is consistent with modeling attribute 4. The operating environments of the systems are implicitly considered, as defined in the event tree logic. However, their numeric values have no connection to the systems’ operational environments.

NASA software conditional risk method [NASA, 2002]- The NASA PRA procedures guide contains a section on assessing software risk, which presented a framework for considering software failures and proposed a conditional risk model for quantifying them. In the conditional risk model, the probability of software-induced system failure can be expressed as the probability of system failure upon the occurrence of a condition which may happen with certain probability. In this representation, the condition can be the reflection of a particular system operating mode, or of the occurrence of an event external to the system of which software is a part, or of a system hardware failure. The probability of occurrence of the condition can be expressed as rate of occurrence multiplied by the mission time duration ( See comment by Guarro in Appendix A). The method was applied in two examples: a spacecraft altitude-control system and a fluid tank control system. In this model, the occurrence of a condition which may trigger a software failure has similarity to the error forcing context (EFC) defined here for modeling software failures. However, in using the model, the conditions are taken into consideration differently than the EFC defined here. The model can be adapted for use in the framework defined with interpretation of the model parameters consistent with the parameters defined here.

Fault injection method [Elks, 2006]- The fault injection method is a useful tool for evaluating the design of digital systems. It can capture the low-level fault-tolerant hardware designs, such as cyclic redundancy check (CRC) as well as the software's normal behavior, not just application software but also platform software and operating system. The limitations of the method include (1) failure to capture all possible ways a digital component/system could fail; that is, stuck-at faults are not the only such ways and (2) its ability to capture software faults, e.g., software design errors, is very limited because too few changes are made to the input values to the software. In this particular application, the method was used to estimate the coverage factors, i.e., the conditional probability that a stuck-at fault is detected, of the components of a digital feedwater control system. The input variables are collected from the plant once per minute for more than 24 hours, i.e., 1440 sets of supposedly *successful* input values were used. The estimated coverage factors then are used in a Markov model [Aldemir, 2006b] that models software in terms of control laws and logic. Accordingly, this model used in this application appears not to be an integrated model of the whole digital feedwater control system. The modular approach, i.e., considering components separately, may have ignored some dependencies in the system.

Traditional Markov model [Swanner, 2000]- The model was developed to aid in generating an update of the ISA standards 84.01 [ANSI, 1996], a companion standard of IEC 61508 [IEC, 2000]. The level of detail of the analysis appears to be consistent with the guidance on Markov models of both standards, i.e., use of safe failure fractions and fault coverages. The Markov model of the Tricon platform distinguishes between safe and dangerous failures of the system, and these two modes were modeled separately to estimate a spurious trip frequency and a probability of failure on demand, respectively. The models were developed for the platform (not an actual application) with assumed input and output modules, and assuming that the software works as expected. The failure rates in the model were estimated from Military Handbook 217. The safe failure fractions and diagnostic coverages were obtained from a proprietary foreign report published by Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology (SINTEF). The author acknowledged that no commonly accepted method exists for quantifying software reliability.

Dynamic Markov model [Aldemir, 2006b]- Unlike the Markov model developed by Swanner [Swanner, 2000], this Markov model is not traditional, and, in addition to hardware component failures, models the control laws and logic of the control system and the control laws of the systems or the plant with which the control system interfaces. The method appears innovative but its application to the digital feedwater control system is not demonstrated. Modeling control laws and control logic is a model of the normal/expected behavior of the software, i.e., what the software is supposed to do, but does not capture possible failures. The level of detail in the model for hardware failure is considered to be similar to that of a traditional Markov model, which models hardware failure and assumes that the software functions as expected. The benefit of the additional detail obtained from the dynamic modeling of the interactions between software, hardware and the plant, needs to be demonstrated. Because of the increase in modeling effort, it would not be surprising if many simplifying assumptions were made to make the modeling feasible.

Dynamic Flowgraph Methodology (DFM) [Aldemir, 2006b]- This is an application of the DFM to the digital feedwater control system. The method [Garrett, 1999] was proposed as one for finding

software faults as well as for modeling software in a PRA. It essentially develops a model for the behavior of the software and integrates it with a model of hardware-component failures of the control system and the part of the plant which interfaces with the software. Its ability to capture software failures depends on how realistically the models represent the software and the respective systems. Each prime implicant generated using the method is a potential challenge (or test) to the software, which may or may not respond properly. In terms of the software-failure modeling attributes, DFM effectively tries to discretize the input space and represent it in terms of a finite number of inputs, while the actual input space may be infinite. The effort needed to develop a detailed model could be prohibitive. Due to the complexity of the software and other systems that the model must cover, it is not possible to identify all software faults, and possibly none could be found, which does not necessarily mean that no fault exists.

Petri nets [Lawrence, 1995] and Formal Methods [Clarke, 1996]- The petri-net method has been used to model the behavior of software [Lawrence, 1995]. Its advantage is its ease of modeling the behavior of a control system. It can be used in modeling changes in the system's state caused by triggering events. For modeling of computer-based systems, the value of Petri nets is that, generally, they can model the state- and time-dependent behavior of computer-based systems. In addition, they can model finite-state machines, concurrent (parallel) processes, software (data flow), communication protocols, synchronization control, and multiprocessor systems. Formal methods are mathematically based languages, techniques, and tools for specifying and verifying design requirements of hardware and software systems [Clarke, 1996]. The process of specification is the act of writing the requirements precisely, thereby giving the developer a deeper understanding of the system specified and the ability to discover design flaws, inconsistencies, ambiguities, and incompleteness. The system can be checked for internal consistency and used to derive other properties of the system. Both petri nets and formal methods are tools developed specifically for modeling the behavior of software and digital systems. Analyses using them could identify software faults and improve the softwares' quality. Their use in modeling software failure in an integrated model of a plant, e.g., a DFM model, should be further explored.

Stochastic Petri net [Balakrishnan, 1996; Malhotra, 1995]- While a petri net describes the state of a machine whose transitions are triggered by discrete events, it does not specify the time spent in any state [Balakrishnan, 1996]. Stochastic Petri-net is a Petri-net whose time of firing is exponentially distributed [Malhotra, 1995]; it is similar to a Markov model and has to be converted into a Markov model to be solved. Therefore, this type of analysis does not appear to significantly differ from that of a Markov model.

#### **C.6.4.2 Review of Software Failure Quantification Methods**

Software-failure quantification methods model a complete software system assuming that the associated digital hardware works as expected. They only consider the operating environment implicitly. These methods were developed for different purposes, and their consistency with our software failure frame work is discussed here.

Most methods reviewed use test data and operational data in the quantification. As discussed, software faults are often corrected once discovered, and the use of software-failure data may not be a good indication of the modified software [Fenton, 2000]. A distinction should be made between test results and operational experience because test environments are not identical to operating ones, and the difference could be the location of failure triggering events. These are the common weakness of the quantification methods.

Individual methods are discussed below.

Bayesian Belief Network (BBN) [Eom, 2004; Dahll, 2002]- In Dahll [2002] a BBN was developed for a computerized system for geographically localizing helicopters. Its main purpose is to aid in a rescue operation if the helicopter makes an emergency landing. The BBN explicitly considers the qualifications of the software producer, quality of the software-production process, the complexity of the problem and its solution and the quality of the product and quality of the analysis. Disparate qualitative information about the software can be included in a single model taking into account the associated uncertainties. It was used to calculate the probability of failure of the system, and used as a tool that supports decision making on the approval of the system. This is consistent with the attributes defined for an actuation system, i.e., the system functions as a safety system that is demanded upon an emergency landing. Using expert elicitation is an advantage of the method, but could be effort- intensive. In Eom [2004], a BBN was developed to assess the quality of the software of a RPS, using as a measure the probability of the “unacceptable” state; this is also consistent with the attributes defined.

Reliability Growth Methods [Schneidewind, 1992; AIAA, 1992] - The reliability growth methods assume that the software’s failure rate changes with time due to detection and correction of software faults during tests and operation. These methods use the experience from both in the same way that hardware-failure data is used to estimate hardware failure rates, to estimate software failure rates as a function of time, by assuming that the latter follows certain formulae whose parameters must be estimated. This type of method models how revisions to software improve its quality. Using a failure rate does not conflict with the attributes defined for a continuously running system (control system). The change in a failure rate corresponds to eliminating some software faults and associated triggering events.

Gray box software reliability [Zhang 2003]- The gray box reliability method uses information on the internal structure of a software to decide on the tests, such that they cover all the nodes and paths, if possible; it employs Bayesian analysis to the test results to estimate the probability that the software would fail in the next execution. The modeling of software failure probability is consistent with the framework discussed in Section C.6. 2. In the case study of the Signal Validation Algorithm that is used for selecting the plant’s instrumentation signal set to constitute the input to the digital Plant Protection System, it was found that not all possible paths of the software were encompassed. All flow paths are effectively treated as if they have identical hardware, and the data analysis follows the standard hardware-component analysis. It is suspected that the same results could be obtained using a much simpler model treating the software as a single component with tests performed on it, and assuming that errors are not corrected.

Metrics Method - In Smidts [2002], the reliability prediction systems based on six software engineering measures were validated, using a personal-access control system (PACS) as a case study. The software reliability is measured by a demand failure probability, and the six measures are mean time to failure, defect density, test coverage, bugs per line of code, function points (a measure of the functional size of the system), and requirement traceability. Using failure probability is consistent with the modeling needs for an actuation system. The University of Maryland's (UMD's) study was reviewed by a panel of experts who deemed the work to be generally sound, meeting the study's objective. Some critical comments include the following: (1) The predictive capability of the method is inadequate for safety critical systems that may have a very stringent requirement on probability of failure on demand and (2) the chosen test cases cannot capture problems of missing requirements.

Software Safety Integrity Level (SIL) of IEC 61508 - IEC 61508 [IEC 61508] is a general standard for safety-related systems that specifies requirements of safety-related systems and provides guidance on assigning SIL. Part 1 of the standard specifies target failure probability for failure on demand and target failure rates for different SILs. Using both failure rate and probability potentially is contradictory, and inconsistent with the attributes defined. For each software SIL, the techniques and measures that are recommended are given in the tables of Annex A and B of Part 3 of the standard; that is, to achieve a certain SIL, appropriate techniques/measures have to be used for life-cycle phases of software. No connection of SILs to the target failure probabilities and rates is provided, the relationship between the SILs and the associated qualitative requirements has to be validated.

#### **C.6.4.3 Review of Methods Currently Used in PRAs**

The applicability of currently used methods for estimating hardware- and human-reliability in modeling software failures in digital systems were assessed. These methods can be easily incorporated within a PRA and satisfy one of the attributes, i.e., they can model software failures in terms of failure rate and/or failure probability. Hardware models primarily are data-driven and many of the human reliability analysis (HRA) models have important features that can address special characteristics of software failures. Considering the available HRA models, the Failure Likelihood Index Method (FLIM) was selected as suitable for modeling software failures in digital systems.



**Table C-5 Scope of Methods and Their Consistency with the Framework**

Modeling Method	1- SW Normal	2- SW Failure	3- HW Normal	4- HW Failure	5- Interaction H+S	6-Context	7-Consistency
Fault tree [Westinghouse 1996]	n	y	implied	y	n	n	y
NASA conditional risk [NASA 2002]	n	y	implied	y	y	partial	y
Fault Injection [Elks 2004]	y	n	y	y (stuck-at faults)	y	n	NA
Markov [Swanner 2000]	implied	n	implied	y	n	n	NA
Dynamic Markov [Aldemir 2006b]	y (control laws)	n	implied	y	y	y	NA
DFM [Garrett 1999]	y	by context	implied	y	y	y	y
Formal methods [Clarke 1996]	y	n	implied	n	n	n	NA
Petri net [Sivertsen 2004]	y	n	implied	n	n	n	NA
Stochastic Petri nets (Balakrishnan)	n	n	implied	y	n	n	NA
BBN [Dahl 2002]	n	y	n	n	n	-	y
Black box [Schneidewind 1992]	n	y	n	n	n	-	y
Gray box [Zhang 2003]	n	y	n	n	n	-	y
Reliability growth [AIAA 1992]	n	y	n	n	n	-	y
Metrics [Smids 2002]	n	y	n	n	n	-	y
SIL [IEC 61508]	n	y	n	y	n	-	n

1. Explicit model of normal behavior of software (implies normal behavior of hardware)
2. Modeling of software failure in terms of a failure rate or probability (questions consistency)
3. Explicit model of normal behavior of hardware
4. Explicit model of hardware failure in terms of a fault, failure rate or probability (implies normal behavior of hardware)
5. Modeling the interactions between hardware and software of digital systems
6. Explicit modeling of the contexts that determine the input to the software
7. Consistency with our basis for modeling software failures

FLIM is an HRA method for qualitative analysis [Chien 1988] and quantification of post-initiator operator actions. FLIM's basic principle is the structured elicitation of expert judgments in the form of ratings and weights for a set of performance-shaping factors (PSFs) that then are used to calculate a dimensionless Failure Likelihood Index (FLI) for each action. The FLI scale is calibrated and converted into an human- error probability (HEP) via a log-linear regression equation. To apply FLIM in modeling software failure, the factors that influence such failures can be defined in terms of PSFs. The examples of PSFs for software include the quality of testing performed, errors identified during testing, testing or use of the software in its particular operational environment, qualifications of the software development team, and number of upgrades performed. With such parameters, a FLI for a software can be defined that, in turn, can be converted to a software failure probability or rate. FLIM has not been used for software reliability assessment and to do so will require systematically developing of PSFs for software, and formulating a process for calculating the ratings and weights of the PSFs.

### **C.6.5 Summary of the Basis for Developing a Software Failure Model**

In this appendix, a framework for modeling software failures in digital systems has been presented considering the different types of such systems used in NPPs. The insights were discussed on the characteristics of software failure apparent from the review, and analyses of actual software failures that occurred in different industries, along with the considerations involved in probabilistic modeling of software failures. Based on the framework defined, a set of attributes that a software failure model should possess were identified. Having such a software failure model is considered to be appropriate for modeling software failures of digital systems in NPPs and incorporating them into a NPP's PRA. Different software failure models that were used in different industries also were studied, considering the attributes defined for use in NPP risk models. The major conclusions for developing a software failure model for digital systems in NPP are summarized below:

1. The operating context of the software, which includes the operating environment and the distribution of the inputs that may be generated in the input space, should be accounted for in developing a software- failure model. Lack of specification of the software's operating context may result in the software failures not being completely defined, and hence, an inadequate model.
2. The concept of EFC triggering a software fault is supported by analyses of experience data. For many software failures, an EFC can be defined and is considered a reasonable way for explaining the failures' occurrence. Both types of EFC, one wherein EFC is the normal operation of the software, and the other in which the triggering event occurs sometime during the operation of the software, are observed in software failures.
3. The occurrence of the EFC is judged to be random, implying that software failures also are random events since they are triggered by the EFC. With this explanation, software failures can be modeled probabilistically. Failure rate and failure probability models can apply, depending on the software's function in the system.

4. Different software failure models have been developed in different industries addressing differing needs and applications. The review of these models is presented, based on a set of attributes defined upon the specific needs for modeling software failures of digital systems in a PRA for NPPs. None of the models can be directly applied to model digital systems for a PRA; some of them are consistent with the attributes defined and can be extended for this purpose. Among the latter, the BBN and the FLIM are considered the best choice for modeling software failures in digital systems.
5. BBN and FLIM can incorporate qualitative information about the software. Without using qualitative information, software reliability can be overestimated since the available data is sparse and, in many cases, sufficient experience in the operating environment/context is unavailable. A specific implementation process, including a process for estimating the parameters in these models, should be developed and tested for software in the digital systems being considered.

## **C.7 Review and Analyses of Software Failures**

Software failures that occurred in many different applications provide important clues for many aspects studied here. Analyzing these failures is essential for understanding the characteristics of these failures and developing models assessing their likelihood and impact on system reliability.

Software failures in nuclear and non-nuclear industries were analyzed to obtain insights on their nature and characteristics. It also afforded useful perceptions for defining a basis for modeling software failures. In this section, analyses are presented of software failure events and insights derived from the results.

### **C.7.1 US Nuclear Power Plant (NPP) Experience**

#### **C.7.1.1 Approach for Experience Data Collection**

Relevant operational events associated with software failures in domestic NPPs were identified to gain knowledge of their nature in terms of characteristics such as the specific cause, the associated EFC, and any dependent failures, such as common-cause failures.

The main approach for identifying software failures in domestic NPPs was conducted via the NRC's "Licensee Event Report (LER) Search System"; the search encompassed the following:

1. The LERs contained in this system that extended from January 1, 1984 through December 31, 2005, a range of 22 years at the time of this study.
2. All plants that operated during this period.
3. All modes of the plants, such as power operation and shutdown.

Since the LER Search System does not directly distinguish failures related to software, a search was made for those LERs containing the keyword “software” in the abstract and title. It yielded 175 LERs. Assuming that if a software failure was significant, it would be mentioned in the LER’s title or abstract, an individual review of each of these 175 LERs would be expected to reveal the most important software failures. Hence, this was considered an efficient way of identifying LERs that potentially are associated with software failures.

These events were complemented with two other sources of events:

1. Volume 2 of NUREG/CR-6734 [Hecht and Hecht, 2001] has 15 events, from 1998 to 2000, that were considered software failures and documented in LERs. The abstract of six of them did not include the keyword “software,” so they were not present in the events obtained from the “LER Search System.” Hence, these additional ones were included in the database (LERs 2061998001, 2201998003, 2751999002, 2821999002, 3021999001, and 3341999011).
2. An additional event involving software failure was known, LER 2931997007, and was added to the database; its abstract did not include the keyword “software.”

The database of the LERs documenting the identified software failure was created using the Microsoft Access database management system. An LER was not included in the database even if the abstract contained the keyword “software” when it was clear that the event did not involve a software failure. For example, this keyword might be used in describing the LER to indicate that the software functioned correctly. Where the description of the LER did not conclusively signify a software failure, the LER was included in the database, and this condition was noted in the description of the event. The review of the 175 LERs resulted in a database of 113 LERs associated, or potentially associated, with software failure(s).

This database is such that it can be sorted by any criteria, such as by the date of the event or by LER number, and searched using one or more keywords in one or more fields. To the extent supported by the information in a LER, each event is characterized in the table in terms of the following properties (fields):

- A) LER Number. An eleven-digit number that uniquely identifies the LER.
- B) Event Date.
- C) Plant. This property indicates the specific nuclear unit(s) involved.
- D) Title. This is the title of the event given by the LER.
- E) Summary. This property describes and identifies the software failure and the associated digital systems are identified. The impact of the software failure on other systems and the overall plant is presented.

- F) Causes. The cause(s) of the software failure. For example, the failure may be due to an incorrect specification of requirements of the software, or to an error during the developmental stage of “Code generation.”
- G) Consequences. The impact of the software failure on the safety of the plant is discussed. It can be characterized in terms of two components, a violation of regulatory requirements and an assessment of the actual safety significance of the overall event.
- H) Error Forcing Context (EFC). This property presents the specific combination of conditions that comprise the EFC, i.e., those that triggered an (inactive) software fault into becoming an (active) software failure.
- I) Dependent failure. This property discusses dependent failure(s) that may have resulted from software failure(s). This involved identifying failures in more than one component, channel, or system.
- J) References. Presently, the only reference is the associated LER.

#### **C.7.1.2 Summary of Results and Insights**

The following results and insights were obtained from the analyses of LER data:

- A) Seventy-one different nuclear units have at least one event related to software failure from January 1, 1984 through December 31, 2005. This means that software failures have occurred in a significant number of units. Hence, this type of failure may occur in any operating units that use software-supported systems.
- B) In 17 of the 113 LERs documenting software failures, two-unit nuclear plants are identified. For example, LER 250-1994-005-02 documents events associated with software failures in Turkey Point Unit 3 and Turkey Point Unit 4. Hence, there have been 130 events associated with software failures in different nuclear units during the period studied.
- C) The forty-five LERs that occurred during the last 10 years of the period in the database, i.e., January 1, 1996 to December 31, 2005, were analyzed to classify the “software failure mode” and the cause of the failure according to the categorization scheme presented in Section C.4. The following conclusions were reached:
  - C.1) Thirty-one out of the 45 events (i.e., about 69%) had the failure mode “Runs with wrong results that is not evident.” The fact that most events studied had this failure mode may be a reason for concern because it is undesirable to have software that is executing, sometimes for long periods, and generating incorrect results. The next most prevalent failure mode, in seven of the 45 events (~16%) was “Runs with evidently wrong results.”

- C.2) Software failures were due to a variety of causes, the most predominant being “Software requirements analysis” with 16 out of the 45 events (about 36%). In general, when software fails this way, it fails to perform a function because when its requirements were specified, the function was not included. The second cause is “Operation and maintenance” with 12 of the 45 events (~27%). Most events related to “Operation and maintenance” involve a failure introduced during modifications or upgrades of the software after it was developed, installed, and had operated for some time. In other words, software that was meeting its expected functions was modified, and some fault was introduced during this process.
- D) Most of the software failures appear to have happened in non-safety-related systems, the main cause(s) of which are not known. Potential reasons are: (1) safety-related systems that use software have higher quality standards, and hence, a lower probability of failure, (2) software may have been more commonly used in non-safety-related systems than in safety-related ones, and (3) a combination of 1 and 2.
- E) In many cases, the specific combination of conditions that comprise the EFC, i.e., the conditions that triggered an inactive software fault into an active one, was identified for a particular LER. The review also revealed that sometimes a failure may occur as soon as the software becomes operational, and may remain hidden for several years. Then, the EFC is the plant’s normal operation. Such failures may be discovered indirectly, such as by discrepancies with the results produced by alternative calculations (see Section C.2, and especially Figure C.2 for a discussion of the EFC). Long-hidden failures, or indirectly discovered ones usually are associated with a non-safety-related system that often has less stringent regulatory requirements than those of safety-related ones.
- F) Most software failures identified in this review had low safety significance for the plant. For example, a software failure might have violated the plant’s regulatory requirements, such as its Technical Specifications. This violation may have resulted in the loss of functionality of some system(s), and an automatic or manual reactor trip. However, during the event, the plant may have had available redundant systems that perform the same function of the lost system(s); accordingly, the safety significance of the software failure may be considered minor. As mentioned earlier, the assessment of the consequence of a software failure on the associated NPP used the LER’s evaluation of its safety impact.
- G) In twenty-nine of the events, i.e., ~ 26% of the 113 LERs, some type of dependent failure occurred, including common-cause failures. Thirteen LERs, i.e., about 12% of the total, potentially involved dependent failures; their LERs did not have enough information to conclusively assess whether such failures had occurred. Hence, the potential of software failures to cause dependent failures, including CCF, is demonstrated. Since a dependent failure can be significant to the risk of an NPP, a software failure also carries this same potential.

## C.7.2 Foreign Nuclear and Non-Nuclear Industry Experience

### C.7.2.1 Approach

The general approach adopted to collect software failure experiences in non-nuclear industries was to search through the internet-based databases. The data search started from websites that briefly described software-related incidents or accidents. Among the events claimed to be a computer-related only a portion could be verified to be caused by computer or software failures from the official reports from different websites, such as FAA, NTSB, NASA, and DOE. Official websites containing databases were further queried to collect more related events. Most of the identified events were verified using the official investigation reports, but a few of the interesting ones were supported by a book or newspaper articles.

Some examples of the non-official websites include “Computer Horror Stories” at <http://www.cs.tau.ac.il/%7Enachumd/horror.html>, “Collection of Software Bugs” at <http://www5.in.tum.de/~huckle/bugse.html>, and Risks Digest at <http://catless.ncl.ac.uk/Risks/>.

The first two list several events with very brief descriptions. The website “Computer Horror Stories” is maintained by Prof. Nachum Dershowitz from the School of Computer Science at Tel Aviv University. The website “Collection of Software Bugs” is maintained by Prof. Thomas Huckle from Institute of Information at TU Munchen. Links to the sources of these events are sometimes provided, but often these links can no longer be accessed. Also, these two websites are not routinely updated. The information in Risks Digest’s web site is updated frequently. It was inaugurated in 1985 and is maintained by Peter G. Neumann (<http://www.csl.sri.com/users/neumann/short.bio>) from the SRI International Computer Science Laboratory. The Risks Forum, known as Comp. Risks in the USENET community, is sponsored by the ACM (Association for Computing Machinery) Committee on Computers and Public Policy (CCPP). The forum illustrates risks to the public from using computer systems and related technology and summarizes, in one line, most of the interesting cases over the past decades. The incidents are not limited to certain area. In fact, either brief or detailed illustrations of almost every important computer-related event can be found here. However, the information in Risks Digest must be further investigated and verified because it usually is not detailed enough and applicability of these stories is not guaranteed.

The sources below contain official reports that were used to verify whether the failure events from the preceding sources were software failures:

1. NTSB Aviation Accident Database at <http://www.nts.gov/ntsb/query.asp>:

The Aviation Accident Database, shared by the NTSB and the FAA, contains data describing the aircraft, operations, personnel, environmental conditions, consequences, probable causes, and contributing factors of civil aviation accidents within the United States, its territories and possessions, and in international waters. An accident is defined as “...an occurrence associated with the operation of an aircraft which takes place between the time any person boards the aircraft with the intention of flight and all such persons have disembarked, and in which any person suffers death or serious injury, in which the aircraft receives substantial damage.” The Safety Board also investigates some incidents,

and includes them in the database in the same form as accidents. An incident is defined as “...an occurrence other than an accident, associated with the operation of an aircraft, which affects or could affect the safety of operations.” The NTSB database website also has an information- query service; some data collected here was obtained e querying it with the keywords “software” and “computer”.

2. ASN Aviation Safety Database at <http://aviation-safety.net/database/>

The ASN Safety Database is updated every week; it has a limited query capability. It briefly describes more than 10,000 safety-related occurrences since 1943 in airliners, military transport category aircraft, and corporate jet aircraft. These descriptions of most events are from official reports often found in the NTSB database.

3. NASA Description of Missions at [http://nssdc.gsfc.nasa.gov/planetary/planetary\\_home.html](http://nssdc.gsfc.nasa.gov/planetary/planetary_home.html)

This database chronologically lists known lunar and planetary missions, including a few historical ones, that were instrumental in the development and evolution of space exploration, covering both successful and failed events. A limited query capability is available. The reports are detailed. Some events collected in the report are verified here.

4. Computer-Related Incidents with Commercial Aircraft at [http://www.rvs.uni-bielefeld.de/publications/compendium/incidents\\_and\\_accidents/index.html](http://www.rvs.uni-bielefeld.de/publications/compendium/incidents_and_accidents/index.html)

This website is maintained by Prof. Peter B. Ladkin’s research group conducting specification, verification, and failure analysis of complex heterogeneous systems. The website gives only incidents and accidents of commercial airplanes, providing for each event, both a brief description and a detailed report. Most of the reports are official but some of the accidents are not related to failure of software or computers.

5. Some other official websites were used that may not maintain a systematic database but contain reports of accident investigations. For example, a 2004 blackout report is available from the DOE’s website.

Other sources of the collected data include various publications and books and are individually referred in each event collected in this report.

### **C.7.2.2 Summary of Results and Insights**

Software failure events were identified in more than 10 different industries, mainly by searching the Internet to identify events, reviewing their descriptions, and screening out those that are not related to software failure, or not considered interesting. Four failure events that took place at foreign NPPs were obtained from a report of the Nuclear Energy Agency [NEA, 1998], and one event at the Davis Besse plant due to a virus [Schulin] that is not reported in the LER database. Forty-three software-failure-related events were identified in non-nuclear industries; including the five nuclear events, 48



events were found. Detailed analyses of selected events are given in the appendix. The consequences of most of the 48 events were very severe because people only tend to identify root causes of very severe events, and only those sources that contain the most important or well-known events are publically available. The nature of this search did not allow the events to be used in a statistical analysis because the screening was subjective, and no attempt was made to identify the period in which the search was performed.

The review of events revealed that software failures occurred in every industry that uses digital systems. Practically all system- and element-level failure modes and failure cause categories defined in this Appendix have taken place. At the software-element level, processing elements fail most often. The more frequent element-failure modes are incorrect implementation and omission of functions or attributes, while errors at a software requirement analysis stage are the most important cause of failure.

Analyses of the software failures support the concept that the occurrence of EFC triggering a software failure is a reasonable way of considering software failures. Accordingly, it is logical to model software failures as random events with probabilities or frequencies. The probability or frequency depends on the software's operating environment, and can be used in reliability modeling of software.

The different types of software failures observed in non-nuclear applications afford useful insights for modeling software failures, in general, which will apply for digital systems in NPPs. The present examples illustrate the needs in modeling levels of detail and in defining a scope of the analysis. For quantitative reliability modeling, the failures do not necessarily have to be modeled explicitly; rather, modeling must need to capture the impacts of the failures and assess the likelihood of software failures.

1. A stuck-at-one fault on a data line of the Traffic Collision Avoidance System of the Korean Air Cargo flight contributed to a near miss collision with British Air flight 027. To capture this type of failure, a model would have to be developed at the individual bit level, and would require a fault injection type of analysis [Cutright 2003].
2. A few events occurred due to faults in diagnostic software, interrupts, and communications; they involve software that is a part of the platform hardware, a part of operating system, and communication software. For example, at Darlington (event 47 in Table C-8), the hardware diagnostic software contributed to stalling the computers and eventually shut the reactor down. To capture these failures, the non-application software must be modeled.
3. A few cyber-security-related events have occurred. Thus, a book [Reed, 2004] written by a former CIA employee reported that a virus the CIA had put in the control software that the Soviet Union's agents purchased caused a major explosion of the natural gas pipeline in Siberia in 1982. Such failure probably is applicable to any software prepared by a vendor and customized by the user. To account for cyber-security-related events, the computer network at a facility has to be included in the model.

4. A few events involved failure of identical software in redundant systems due to CCSFs. For example, a software exception caused failure of both initial reference systems of the Ariane 5 launch vehicle that exploded during a takeoff [Lions]. CCSF is real and has to be modeled in a quantitative reliability model.
5. Poor human-machine interfaces contributed to a few accidents. Thus, in the 2003 blackout, the computer alarm system at First Energy was unavailable for a long time without any indication due to a race condition [Jesdanun 2004], and this prevented early mitigation of the blackout. In the case of China Airline's flight A300 [Greenwell], a conflict between an autopilot and a human pilot caused the plane to crash. A good model of operator behavior along with software behavior would be needed for identifying this accident scenario.

### **C.7.3 Insights on Modeling Software Failures in Digital Systems for NPPs**

Analyses of experience data for nuclear and non-nuclear industries provide useful insights that supported the concepts of software failures and the probabilistic modeling of these failures. Table C-6 breaks down the software failures observed in different industries in terms of the system's failure modes and causes. Here, specific insights derived for modeling software failures are summarized.

1. The assessment of operating experience shows that software failures occur in different systems that use software and a "system-centric" view of software failure, as used in this report.
2. Software failure modes and failures causes defined in this Appendix are consistent with, and sufficient for, categorizing the failures observed for different applications. A new software failure mode introduced here, i.e., software runs with incomplete or incorrect display of information or misleading command to the user, is based on the analyses of experience data.
3. The concept of EFC triggering a software failure is supported by the analyses of experience data. For many software failures, an EFC can be defined and is considered a reasonable way for explaining the failure's occurrence. Both types of EFC are observed: one where EFC is the normal operation of the software, and the other wherein the triggering event occurs sometime during the operation of the software.
4. The occurrence of the EFC is judged to be a random failure, implying that software failures also are random since they are triggered by the EFC. Under this explanation, software failures, can be modeled probabilistically.

**Table C-6 Software Failures Modes and Causes Observed in Nuclear and Non-Nuclear Industry Experience Data<sup>1</sup>**

<b>Industry</b>	<b>Software Failure Mode</b>	<b>Percentage of Observed Failures<sup>2</sup></b>	<b>Software Failure Causes</b>	<b>Percentage of Observed Failures<sup>3,4</sup></b>
Nuclear	SFM-1	2	SFC-I	0
	SFM-2	9	SFC-II	36
	SFM-3	16	SFC-III	7
	SFM-4	69	SFC-IV	0
	SFM-5	0	SFC-V	2
	SFM-6	0	SFC-VI	27
			SFC-VII	0
			SFC-VIII	4
			SFC-IX	0
			SFC-X	0
Non-Nuclear	SFM-1	2	SFC-I	2
	SFM-2	2	SFC-II	46
	SFM-3	54	SFC-III	23
	SFM-4	33	SFC-IV	6
	SFM-5	4	SFC-V	4
	SFM-6	2	SFC-VI	3
			SFC-VII	10
			SFC-VIII	14
			SFC-IX	6
			SFC-X	0

Notes on Table C-6:

1. The categorization of data in this table is intended for obtaining insights on the characteristics of software failure. It is not intended for statistical analysis or for developing parameters for software reliability models. In this analysis, 45 events are included for US nuclear industry, and 48 for non-nuclear industry.
2. Percentages do not necessarily add up to 100 because adequate information was not available in all cases to define the categorization. SFM categorization could not be developed for 4% of the data in the nuclear industry, and 2% in non-nuclear industry.
3. Percentages do not necessarily add up to 100 because adequate information was not available in all cases to define the categorization. SFC categorization could not be developed for 24% of the data in the nuclear industry, and 4% in non-nuclear industry.
4. For data in non-nuclear industry, more than one cause may be defined for a particular event. Accordingly, the sum of the percentages is larger than 100.

## **C.8 Software Failures: Detailed Data**

### **C.8.1 Software Failures in U. S. Commercial Nuclear Power Plants (NPPs)**

The software failures identified for commercial NPPs operating in the United States are listed in Table C-7. For each event, the date of event occurrence, the plant where the event took place, LER number, and a title is given.

**Table C-7 Events Related to Software Failures in US NPPs**

<b>LER Number</b>	<b>Event Date</b>	<b>Plant</b>	<b>Title</b>
1551988007	June 29, 1988	Big Rock Point	Technical Specification Shutdown - Inoperable Neutron Monitoring System
2061998001	January 14, 1998	San Onofre 1	The title is not known. (The Licensee Event Reports that are categorized as "Safeguards/Security" are not available for general public viewing.)
2201984009	July 13, 1984	Nine Mile Pt 1	Both Fuel Zone Water Level Monitoring System Channels Inoperable Simultaneously
2201986007	April 15, 1986	Nine Mile Pt 1	Software Error Causes Inaccurate Fuel Zone Level Indication
2201986019	June 18, 1986	Nine Mile Pt 1	Loss of RWM During Start-Up With Less Than 12 Rods Withdrawn and Less Than 20% Power
2201986034	December 6, 1986	Nine Mile Pt 1	Loss of Stack Sample Flow Due to Software Problem
2201998003	March 4, 1998	Nine Mile Pt 1	Power/Flow Relationship Technical Specification Violation (Operation Above Rated Power) Due to Inadequate Managerial Methods
2201998018	October 6, 1998	Nine Mile Pt 1	Violation of License Condition 2.C.(1) and the Power/Flow Relationship Technical Specification Due to a Degraded Valve
2471999019	October 28, 1999	Indian Point 2	Inadvertent Disabling of Rod Position Plant Computer Program
2501994005	November 3, 1994	Turkey Point 3, Turkey Point 4	Design Defect in Safeguards Bus Sequencer Test Logic Places Both Units Outside the Design Basis
2541993008	July 9, 1993	Quad Cities 1, Quad Cities 2	U1 and U2 Rx Bldg. Vent Radiation Monitors Hi Hi Setpoint Non-conservative Due to Calibration Error Caused by An Errant Computer Program Used to Calculate Source Exposure.
2541994017	December 5, 1994	Quad Cities 1, Quad Cities 2	Banked Position Withdrawal Sequence Rules Violated Since October of 1991 Due to, Training, Procedure and Work Practice Deficiencies in The Nuclear Engineering Group.
2601996005	May 10, 1996	Browns Ferry 2	Unit 2 Scrammed on Low Reactor Water Level Due to The Digital Feedwater System Reinitializing Its Feed Pump Demand Output Signal to Zero and Subsequent Trip of The Reactor Core Isolation Cooling on High Exhaust
2602001003	July 25, 2001	Browns Ferry 2	Automatic Reactor Scram Due to a Turbine Trip During Routine Testing
2602004001	July 8, 2004	Browns Ferry 2	Reactor Scram from Sensed Power Load Unbalance Condition

<b>LER Number</b>	<b>Event Date</b>	<b>Plant</b>	<b>Title</b>
2611992023	November 18, 1992	Robinson 2	Failure of ERFIS Processing Function Results in Inoperability of Control Rod Monitoring Function
2631986010	May 8, 1986	Monticello	Standby Gas Treatment System Initiation During Wide Range Gas Monitor Source Check
2701998004	July 16, 1998	Oconee 2	Technical Specification Snubber Surveillance Interval Exceeded Due to An Inadequate Process
2721998004	March 2, 1998	Salem 1, Salem 2	Failure to Perform Radioactive Effluent Concentration Surveillance and Effluent Monitoring Instrument Channel Setpoint Determinations in Accordance With Technical Specifications
2751990014	December 5, 1990	Diablo Canyon 1	Reactor Trip on Turbine Trip Due to Inadequate Evaluation of Runback Limit Setpoint
2751992028	October 1, 1992	Diablo Canyon 1, Diablo Canyon 2	Technical Specifications 3.3.3.8 and 3.7.10 Not Met Due to Procedural Deficiency
2751999002	March 20, 1999	Diablo Canyon 1	Technical Specification 3.3.1 Not Met Due to Inadequate Knowledge and Communication
2781995006	October 25, 1995	Peach Bottom 3	Technical Specification Violation Due to Exceeding Licensed Reactor Power Due to Calculation Software Problem
2801986011	February 27, 1986	Surry 1	Inoperable Hi Range Radiation Monitors
2821999002	January 8, 1999	Prairie Island 1	While at Hot Shutdown a RCP was Tripped During Surveillance Testing of RCP Breakers, Resulting in no RCPs Running and an Auto-Start of an Auxiliary Feedwater Pump
2861993005	December 31, 1992	Indian Point 3	Missed Periodic Inservice Tests and Faults in AMSAC System Logic, Due to Personnel Error, Place the Plant Outside Design Basis
2931997007	April 1, 1997	Pilgrim	Safeguards Buses De-Energized and Losses of Off-Site Power During Severe Storm While Shut Down
3021999001	January 18, 1999	Crystal River 3	Personnel Failed to Perform a Surveillance Requirement Within the Time Specified in The Improved Technical Specifications With a Computer Alarm Inoperable

<b>LER Number</b>	<b>Event Date</b>	<b>Plant</b>	<b>Title</b>
3051997003	March 10, 1997	Kewaunee	Plant Operation Outside of Test Specs with Reactor Vessel Level Indication Out of Service
3151986010	May 15, 1986	Cook 1	ESF Actuation Caused by a Radiation Monitor Software Error
3151998015	March 12, 1998	Cook 1, Cook 2	Ice Weight Requirements Potentially Not Met Due to Nonconservative Assumption in Software Program
3161984003	March 11, 1984	Cook 2	Actuation of an Engineered Safety Feature
3161984008	April 19, 1984	Cook 2	Containment Purge Isolation Resulting From Radiation Monitor Software Errors
3161984011	May 4, 1984	Cook 2	Containment Purge Isolates due to Software Errors
3161987014	December 15, 1987	Cook 2	Missed Surveillance Due to Personnel Error in Process Computer Software Programming
3162000007	June 28, 2000	Cook 2, Cook 1	Technical Specification 3.0.3 Shutdown Initiated Due to Inoperable Rod Position Indications
3171993002	February 5, 1993	Calvert Cliffs 1, Calvert Cliffs 2	Missed Surveillance Requirements Due to Software Manual Error
3211984008	April 30, 1984	Hatch 1	Missed Surveillance
3211989003	March 2, 1989	Hatch 1	Tracking Program Software Deficiency Results in Missed Surveillance
3211989017	November 28, 1989	Hatch 1, Hatch 2	Personnel Error Results in Incorrect Liquid Radwaste Discharge Monitor Setpoint
3241989005	March 13, 1989	Brunswick 2, Brunswick 1	HPCI Declared Inoperable Due to a Limitorque Corporation Software Error
3252005001	April 9, 2005	Brunswick 1, Brunswick 2	Operation Prohibited by Technical Specification - Inoperable Feedwater and Main Turbine High Water Level Trip
3312001005	October 3, 2001	Duane Arnold	Licensed Power Level Exceeded Due to Use of Non-conservative Constant in Heat Balance
3331990029	December 7, 1990	Fitzpatrick	Suppression Pool Average Water Temperature Monitor Inoperable Due to Engineering Error Entering Incorrect Criteria Into Instrument Software Program

<b>LER Number</b>	<b>Event Date</b>	<b>Plant</b>	<b>Title</b>
3331995015	December 15, 1995	Fitzpatrick	Omission of RWR Seal Purge Flow From Reactor Heat Balance
3341997036	November 13, 1997	Beaver Valley 1, Beaver Valley 2	Inadequate Channel Check for Meteorological Monitoring Instrumentation
3341999011	September 9, 1999	Beaver Valley 1, Beaver Valley 2	Inadequate Axial Flux Difference (AFD) Monitor Alarm Surveillance
3351986005	June 20, 1986	St Lucie 1	Technical Specification Deviation Due to Personnel Error
3351997002	February 21, 1997	St Lucie 1	Operation in Excess of Maximum Rated Thermal Power Due to Digital Data Processor Calorimetric Error
3362000013	July 31, 2000	Millstone 2	Failure to Calculate Azimuthal Power Tilt As Required by Technical Specifications
3381986003	February 20, 1986	North Anna 1	Failure of Radiation Monitors Due to Software Design and Other Problems
3382000001	February 21, 2000	North Anna 1	Control Rod Deviation Monitor Inoperable Due to Personnel Error
3411997001	February 6, 1997	Fermi 2	Error in Mass Flow Conversion Algorithm in the Heat Balance Methodology for Calculating Core Thermal Power
3412004004	December 4, 2004	Fermi 2	Automatic Reactor Shutdown Due to Automatic Voltage Regulator Failure
3441987020	August 3, 1987	Trojan	Seismic Monitoring Instrumentation Surveillance Missed Due to Inadvertent Deletion From Schedule
3441989028	August 1, 1989	Trojan	Personnel Error in Preparing Procedure Results in Missed Rod Position Surveillance
3461988006	February 22, 1988	Davis-Besse	Software Error in Kaman Radiation Monitors
3482000006	May 28, 2000	Farley 1	Reactor Trip from 4% Power Due to Personnel Error
3531998005	June 29, 1998	Limerick 2	Tech Spec (TS) Violation in that a Surveillance Test Exceeded its TS Surveillance Period due to Personnel Error & a Weakness in the use of the Scheduling Program
3611986018	July 7, 1986	San Onofre 2	Unit 2 Trip Due to Failure of Control Rod Drive System
3622000002	February 1, 2000	San Onofre 3	Missed RCS Leak Rate Surv. - Y2K Error Inattention to Detail



<b>LER Number</b>	<b>Event Date</b>	<b>Plant</b>	<b>Title</b>
3661997006	April 7, 1997	Hatch 2	Data Entry Error Results in Missed Technical Specifications Surveillance on Source Range Monitors
3681994003	August 24, 1994	Arkansas 2	Control Element Assembly Position Indication Surveillance Testing Not Performed as Required by Technical Specification Due to Personnel Error Associated With Computer Software Change
3681997004	May 26, 1997	Arkansas 2	Alternate Radioactive Gaseous Effluent Sampling Not Established Within One Hour As Required Due to Inadequate Alarming Capabilities on Radiological Dose Assessment Computer System Terminals
3681998003	May 20, 1998	Arkansas 2	Surveillance Testing of Control Element Assembly Position Indication Maximum Deviation Was Not Performed for One Group As Required by Technical Specifications Due to Inadequate Verification Or Validation of A Computer Software Change
3691994008	November 1, 1994	McGuire 1	Three Auxiliary Feedwater to Steam Generator Isolation Valves Were Determined to Be Past Inoperable Due to A Fabrication Deficiency in Vendor Supplied Testing Software.
3732000003	June 24, 2000	LaSalle 1, LaSalle 2	P-Bypass Setpoint Set Non-Conservatively Due to Inappropriately Turning Off the Process Computer Feedwater Flow Density Correction Program
3741984049	August 1, 1984	LaSalle 2	Reactor Radiation Doors Unsecured
3821986025	October 22, 1986	Waterford 3	Reactor Trip During Startup Due to Prolonged Low power operations
3821989012	July 6, 1989	Waterford 3	Radiation Monitor Inoperable during Discharge due to Inadequate Administrative Controls
3821990016	October 14, 1990	Waterford 3	Improper Access Control to A High Radiation Area Due to A Security Computer Software Inconsistency
3821992001	January 22, 1992	Waterford 3	Failure to Satisfy Technical Specification Surveillance Requirement due to Inadequate Administrative Controls and Inadequate Attention to Detail
3821994018	November 23, 1994	Waterford 3	Liquid Radioactive Waste Released from Waste Condensate Tank 'A'
3821997013	April 7, 1997	Waterford 3	Refueling Machine Failed to Meet Technical Specification Requirements

<b>LER Number</b>	<b>Event Date</b>	<b>Plant</b>	<b>Title</b>
3871997001	January 2, 1997	Susquehanna 1, Susquehanna 2	SPING Terminals Not Communicating With Field Units
3951984013	March 9, 1984	Summer	Smoke Detectors Temporarily Inoperable
3951990003	April 6, 1990	Summer	Computer Software Error Caused Nonconservative Radiation Monitor Setpoints
3951992001	January 11, 1992	Summer	Missed Surveillance for Axial Flux Difference
3971996004	June 24, 1996	Columbia	Manual Reactor Scram Due to Digital Feedwater System Error Found During Testing
4121996001	February 12, 1996	Beaver Valley 2	Condition Prohibited by Technical Specifications, Missed Rod Position Surveillance
4131986017	March 17, 1986	Catawba 1	Catawba Nuclear Station, Unit 1
4141987008	March 3, 1987	Catawba 2	Technical Specification Violation Due to A Design Deficiency in the Operator Aid Computer Reactor Coolant Leakage Program
4231986011	February 5, 1986	Millstone 3	Control Building Isolation Signals Due to Noise Spike
4231989029	November 20, 1989	Millstone 3	Missed Axial Flux Difference Technical Specification Surveillance Due to Procedural Inadequacy
4241987056	September 16, 1987	Vogtle 1	Technical Specification Not Met Due to Incomplete Vendor Software for Dose Calculations
4241987058	September 21, 1987	Vogtle 1	False Signal From a Radiation Monitor Leads to Control Room Isolation
4241987065	November 9, 1987	Vogtle 1	Containment Ventilation Isolation Due to Actuator Failure and Software Design
4241987068	November 17, 1987	Vogtle 1	Control Room Isolation Due to Faulty Sensing Tube and Software Design
4241987073	December 21, 1987	Vogtle 1	Containment Ventilation Isolation Due to Sensing Tube Failure and Software Design
4241988030	October 27, 1988	Vogtle 1	Surveillance Missed Due to Inoperable Rod Position Deviation Monitor
4241988038	November 16, 1988	Vogtle 1	Erroneous Neutron Detector Indicators Lead to Plant Operation Outside of Tech. Spec.

<b>LER Number</b>	<b>Event Date</b>	<b>Plant</b>	<b>Title</b>
4401999007	December 16, 1999	Perry	Operating License Thermal Power Limits Exceeded During Previous Cycle Coastdown
4402000002	March 1, 2000	Perry	Inadequate Data Validation Checks Result in Missed Power Distribution Limits Surveillance Requirements
4431994018	November 28, 1994	Seabrook	Missed Technical Specification Surveillance Requirement
4541984024	November 29, 1984	Byron 1	Loss of Control Room Annunciation of Radiation Monitors
4541986014	May 12, 1986	Byron 1	Control Room Ventilation Actuation Due to High Vacuum Alarm on OPR32J Radiation Monitor
4551987011	July 25, 1987	Byron 2	Reactor Trip during 30% Load Rejection Test on Overtemperature Delta T Due to Steam Dumps Failure to Fully Open and the Digital Electro- hydraulic Sequential Valve Mode
4552005001	October 19, 2005	Byron 2, Byron 1	Unit 2 Automatic Reactor Trip Due to Low Steam Generator Level resulting from a Software Fault on the Turbine Control Power Runback Feature
4561987057	October 9, 1987	Braidwood 1	Turbine Trip and Subsequent Rx Trip During Monthly Turbine Valve Cycle Surveillance
4582001002	October 4, 2001	River Bend	Potential Violation of Maximum Power Limit Due to nonconservative Error in Core Thermal Power Calculation Software.
4821986045	September 2, 1986	Wolf Creek	Moderate Loss of Physical Security Effectiveness Due to Computer Malfunction
4821999014	December 2, 1999	Wolf Creek	Computer Leak Rate Calculation for Containment Sump Leakage Indication Does Not Meet Design
4822000002	May 24, 2000	Wolf Creek	Loss of Containment Total Unidentified Leak Rate Computer Point Operability
4831986039	October 19, 1986	Callaway	Action Statement Not Entered When Less Conservative Radiation Monitor Setpoint Calculated Due to Computer Software Error
4831999009	December 3, 1999	Callaway	RCS Leakage Detection Systems are Outside of Design Basis Because a 1 gpm Leak cannot Be Detected within 1 Hour
4981988014	February 4, 1988	South Texas 1	Reactor Protection System Actuation Due to a Software Problem in QDPS

<b>LER Number</b>	<b>Event Date</b>	<b>Plant</b>	<b>Title</b>
4981989016	July 13, 1989	South Texas 1	Technical Specification Violation Due to Inadequate Procedural Control Over a Plant Modification
4991990016	October 27, 1990	South Texas 2	Fuel Handling Building HVAC Actuation Due to Loss of Power
5281986046	June 25, 1986	Palo Verde 1	Incorrect Computer Constants Render the Containment Radiation Monitor Inoperable
5281992011	July 13, 1992	Palo Verde 1, Palo Verde 2	Missed Technical Specification Action for COLSS Inoperable
5291986017	September 11, 1986	Palo Verde 2	Reactor Trip Initiation by Reactor Protection System
5291998002	February 28, 1998	Palo Verde 2	Missed Core Protection Calculator Shift Channel Check due to Data Link Failure
5292005004	August 22, 2005	Palo Verde 2	Technical Specification Required Shutdown due to Core Protection Calculators Inoperable

## **C.8.2 Software Failures in Foreign Nuclear and Non-Nuclear Industries**

Software failures identified for foreign NPPs and non-nuclear industries are presented in Table C-8. For each of the event, the following information is included: a title, the date of occurrence, the EFC, system failure mode (SFM), element failure mode (EFM), and the system failure cause (SFC).

## **C.8.3 Detailed Analyses of Selected Software Failure Events**

A detailed analysis was conducted of each of the software failures identified in this report. The detailed analysis was used to determine if an error forcing context (EFC) contributed to the software failure, and to identify the system failure mode (SFM), the element failure mode (EFM), and the software failure cause (SFC). Detailed analyses of the following 11 selected events are presented in this appendix.

- Overdose of Radiation Therapy Machine THERAC-25 (1985-1987)
- London Ambulance Dispatch System (October 1992)
- China Airline Flight B1816 Crash at Nagoya (April 26, 1994)
- Turkey point Diesel generator Sequencer (November 4, 1994)
- Common Cause Failure of Voltage Regulating Transformers and Vital AC Buses at Pilgrim (April 1, 1997)
- Core Protection Calculators Inoperable at Palo Verde 2 (August 22, 2005)
- Slammer Virus in Davis-Besse Nuclear Power Plant (January 25, 2003)
- Natural Gas Pipeline Explosion in Soviet Union (Summer 1982)
- Maroochy Water Treatment Plant Accident (2000)
- Blackout of North America (August 14, 2003)
- Common Cause Failure of Security Computers at San Onofre Unit 1(1998)

The following information is obtained from the detailed analysis of each of the software failure event: a summary of the event, the software failure in the event, a brief summary of the consequences of the software failure, an assessment of the EFC that resulted in the software failure, failure categorization in terms of SFM and SFC, whether the failure involved dependent or common cause failure, and any applicable discussion.

**Table C-8 Summary of Software Failures in Foreign NPPs and Non-Nuclear Industries**

Event	Date	Error Forcing Context (EFC)	System Failure Mode	Element Failure Mode	Failure Cause
1. Theratron780-C Teletherapy Unit at Panama's National Cancer Institute	Aug. 2000 - Mar. 2001	Entering data of multiple shielding blocks together as a single block	SFM-4	EFM-1.3 EFM-2.3	SFC-II-1 SFC-II-2
2. Overdose of Radiation Therapy Machine Therac-25	1985 - 1987	Selecting "x-ray" treatment and quickly correcting it to "electron" treatment in 8 seconds	SFM-4	EFM-1.1	SFC-II-1 SFC-V-1 SFC-V-2
3. London Ambulance Dispatch System Failure	Oct. 26, 1992	Unavailable correct vehicle status and location	SFM-3	EFM-1.6 EFM-2.6 EFM-5.5	SFC-I-1 SFC-V-2
4. 2003 August Blackout in North American	Aug. 14, 2003	A unique combination of events causing the stalling and queuing	SFM-2	EFM-2.1 (or EFM-5.1) EFM-2.3	SFC-II-2 SFC-III-7
5. Southwest Airline Flight 1565 No. 2 Engine Fire Incident	Jul. 7, 1998	Unstable sine voltage output of Channel B	SFM-3	EMF-1.3	SFC-II-2
6. N954VJ Collision Accident in Charlotte Airport	Jul. 2, 1994	Presence of wind shear and flaps in transition when landing	SFM-4	EFM-5.5	SFC-III-2
7. Uncommanded Rolls of N331NW	Apr. 27, 1995	Disrupted input (voltage spike) to ELAC	SFM-3	EFM-1.3	SFC-II-2
8. Midair Collision Incident of N1801B	Oct. 17, 2000	Incorrect transponder setting on airplane that is in range of tower	SFM-5	None	SFC-II-2 SFC-VII-2
9. FedEx MD11 N611FE Crash During Landing at Newark International Airport	Jul. 31, 1997	Overcontrol of airplane during landing	SFM-4	EFM-5.5	SFC-III-2
10. An Almost Midair Collision of BA027	Jun. 28, 1999	Incorrect flight altitude data sources	SFM-4	EFM-1.3	SFC-II-2
11. China Airline B1816 Crash at Nagoya Airport	Apr. 26, 1994	Pilot attempts to control the aircraft without knowing that the autopilot is engaged	SFM-5	EFM-5.4 EFM-2.3	SFC-II-2 SFC-II-3

Event	Date	Error Forcing Context (EFC)	System Failure Mode	Element Failure Mode	Failure Cause
12. French Airbus Flight 148 Crash Near Strasbourg	Jan. 20, 1992	Wrong selection of one of the descent modes.	SFM-5	None	SFC-VIII
13. A330 Test Flight Crash at Blagnac Airport, Toulouse	Jun. 30, 1994	Loss of an engine and cut-off of the hydraulic with settings of altitude less than 7,000 feet.	SFM-3	EFM-5.5	SFC-II-3
14. Airbus A320 Crash at Air Show Near Mulhouse	Jun. 26, 1988	When the flight switches from one flight mode to another	SFM-4	EFM-1.5	SFC-II-X Or SFC-IV-X <sup>2</sup>
15. Korean Air Flight 801 Crashed into Nimitz Hill in Guam	Aug. 6, 1997	When the airplane needs MSAW (when the altitude is too low) information in 54-nm-inhibition range	SFM-4	EFM-2.3	SFC-VIII
16. Ariane 5 Failure 40 Seconds After First Launch	Jun. 4, 1996	Too high horizontal velocity in the lift-off	SFM-4	EFM-5.5	SFC-II-1
17. Mars Climate Orbiter (MBO) Disappeared After Maneuvering into Target Martian Orbit	Sep. 23, 1999	When the thruster firings are performed	SFM-4	EFM-5.5 or EFM-2.5	SFC-III-1
18. Patriot Missile Failure to Track and Intercept a Scud Missile	Feb. 25, 1991	The up time of the system is too long	SFM-4	EFM-5.5	SFC-III-1 SFC-VIII
19. Swedish Gripen Fighter Crash on Landing in Sixth Test Flight	Feb. 2, 1989	Stick movements exceed the predicted values at low speed	SFM-4	EFM-5.4	SFC-III-2
20. A320 Incident Due to Spoiler in Maintenance Mode	Aug. 26, 1993	Spoilers are in maintenance mode in one side of the plane after taking-off	SFM-3	EFM-1.3 or EFM-5.3	SFC-II-1
21. U.S. Navy Smart Ship Failure of Propulsion System	Sep., 1997	Enter zero (or other values), which can cause the database to overflow, into the data field	SFM-3	EFM-1.3	SFC-II-1 C-II-2

---

<sup>2</sup> “X” is used due to insufficient information.

Event	Date	Error Forcing Context (EFC)	System Failure Mode	Element Failure Mode	Failure Cause
22. Pathfinder Reset Problem Caused Loss of Data From Mars	Jul., 1994	When medium priority task is scheduled while the high priority task is blocked waiting for ASI/MET task	SFM-3	EFM-4.5	SFC-II-1
23. Mariner 1 Destroyed After Launch Failure Observed	Jul. 22, 1962	Malfunction of the airborne beacon. (In this situation, the incorrect guidance signals were allowed to transmit)	SFM-3	EFM-1.5	SFC-IV-1
24. AT&T Telephone Outage in 1990	Jan. 15, 1990	Recovery of a heavily loaded 4ESS switch from temporary failure, which causes more than one call to the connecting 4ESS switches in very short period	SFM-3	EFM-2.5	SFC-II-1
25. Telephone Outage in 1991	Jul. 1 - 2, 1991	Too many computer-generated messages for the DSC communication software	SFM-3	EFM-5.3	SFC-VI-1 SFC-II-2
26. Apollo 8 Positioning Problem Due to Accidental Entry to Computer	Dec., 1968	Data stored in certain area of the memory was erased by accidentally entering wrong data entry	SFM-3	EFM-5.3 EFM-5.4	SFC-II-1
27. NORAD False Alarm of Nuclear Attack in 1980	Jun. 3, 1980	The failure of the computer chip	SFM-4	None	SFC-IX
28. Gemini 5 Landing with 169 km Off Target	Aug. 21, 1965	Whenever the spacecraft re-enters the atmosphere using the ground based computer guidance program.	SFM-3	EFM-5.3	SFC-III-2
29. Delta III Launch Failure Due to Design of Control System Software	Aug. 26, 1998	Development of the 4 Hz roll mode that will be triggered as long as the Delta III is launched	SFM-3	EFM-5.4	SFC-III-2
30. Titan IV Put Satellite into Incorrect Orbit Due to Loss of Control in Launch	Apr. 30, 1999	Lift-off of Titan IV	SFM-4	EFM-5.6	SFC-VIII
31. Phobos I Incident Due to Depleted Solar Array Batteries Caused by Software Error	Sep. 2, 1979	The letter that was left out in the message uploaded to Phobos I from control center and it was needed during the transfer between two command centers	SFM-3	EFM-5.4	SFC-VIII
32. An Inmate Escaped Due to Faulty Computer System	Spring, 1992	The second computer is busy when the first one calls	SFM-3	EFM-3.5 EFM-5.5	SFC-II-3



Event	Date	Error Forcing Context (EFC)	System Failure Mode	Element Failure Mode	Failure Cause
33. NY Stock Exchange Opened Late Due to Communication Problem in Software	Dec. 18, 1995	Undefined due to insufficient details	SFM-3	EFM-3.X	X
34. EFTPOS Crash Due to Undistributed Workload of a Failed Processor	June 2, 1997	Failure of a processor	SFM-3	EFM-4.X	X
35. Sewage Spill in Willametter River Caused by Alarm System Software Failure	Sep. 19 and 20, 1988	Sewage treatment pumps stop working	SFM-3	EFM-5.5	SFC-IV-X
36. Sewage Spill in Willametter River Caused by Loss of Electric Power	June 7, 1988	Loss of electric power for computer	SFM-3	None	SFC-VII-3
37. Slammer Virus in Davis-Besse Nuclear Power Plant	Jan. 25, 2003	External consultant linked to plant's intranet and caused the infection of slammer worm of a computer that is interconnected to internal network of the plant with MS-SQL server without installing the security patch	SFM-3	EFM-1.6 EFM-5.6	SFC-VI-3 SFC-VII
38. Maroochy Water Treatment Plant Accident Caused by Hacking	Early, 2000	When the water treatment plant is attacked by a person who is either an insider or knowledgeable enough to attach the plant's control system remotely	SFM-3	EFM-2.3 EFM-3.3	SFC-II-2 SFC-VII
39. Gazprom Hacked	2000	Hacker's program embedded in benign code is not identified by the operators	SFM-4	EFM-X.4	SFC-VII
40. Roosevelt Dam Hacker	1998	Undefined due to insufficient information	SFM-4	X	SFC-VII
41. Natural Gas Pipeline Explosion in Soviet Union	Summer of 1982	Faulty software was not identified after certain time interval of installation	SFM-4	EFM-X.5	SFC-VII
42. F-14 Weight on Wheels	Unknown	Pilot decides to raise landing gears while the airplane is still on the ground	SFM-3	EFM-5.5	SFC-II-1 SFC-II-2
43. Train Signal System Software Problem	July 1990	Undefined due to insufficient details	SFM-5	EFM-5.5	SFC-III-X
44. Train Door Failure	Jan. 8, 2000	When the door of the train is not closed properly	SFM-3	None	SFC-IX

<b>Event</b>	<b>Date</b>	<b>Error Forcing Context (EFC)</b>	<b>System Failure Mode</b>	<b>Element Failure Mode</b>	<b>Failure Cause</b>
45. Bruce Refueling Accident	Jan. 23, 1990	Software bug in the refueling machine code	SFM-3	EFM-5.2	SFC-III-5
46 Software Patch Caused Reactor Trip	Jan. 15, 1992	A software patch used during shutdown to bypass rationality check was not removed, one sensor was down for maintenance, and a second one had a faulty diode	SFM-3	EFM-1.6	SFC-VI-4
47. Diagnostics Caused Reactor Trip	-	An engineer requested longer than expected data causing delay in hardware checking software	SFM-1	EFM-4.1	SFC-II-1
48. Memory Failure Caused Pump Trip	Feb. 23, 1996	1. Memory failure caused recirc. Pump trip 2. Incorrect control constants being transferred from the malfunctioned memory element to the replacement memory element	SFM-3	1. EFM-5.5 2. --	1. SFC-II-1, SFC-II-3 2. SFC-VIII

### **C.8.3.1 Overdose of Radiation Therapy Machine THERAC-25 [Leveson 1993, Peterson 1995] - 1985-1987**

#### **C.8.3.1.1 Summary**

THERAC-25 is a computerized radiation therapy machine manufactured by Atomic Energy of Canada Limited (AECL). It had been installed and operated at 11 different medical facilities during 1985-1987. It has two methods of treating patients, using accelerated electrons and x-rays generated by hitting a target with high energy electrons. The overdose accidents occurred when the high energy electrons used to generate x-rays were mistakenly used directly on patients, due to software errors.

#### **C.8.3.1.2 Software Failures**

The software of THERAC-25 was developed by a single person using PDP 11 assembly language over a period of several years. The software evolved from that of an earlier version of Therac. There appears to be more than one software failures/bugs, because the “the investigators could not reproduce the fault condition that produced the 1987 Yakima overdose” [Leveson 1993]. The one identified was a race condition<sup>3</sup> that was not considered in the software. When an operator first selected “x-ray” treatment by mistake and quickly corrected it to “electron” treatment in less than 8 seconds (this is the error forcing context), a part of the software is not aware of the correction, and high energy electron is used instead of the lower dose rate specified by the operator. High energy electrons are used to hit a target to generate x-rays. Much lower energy electrons are used for direct treatment of patients. Other contributors include removal of hardware interlock, unacceptable software engineering practice (no documentation, inadequate testing), poor user interface, and accident management inadequacy.

#### **C.8.3.1.3 Consequence**

6 patients overdosed and 4 of them died. FDA declared THERAC-25 defective. Law suits were brought upon by families of patients.

#### **C.8.3.1.4 Likelihood of Error Forcing Context**

According to “Fatal Defect” [Peterson 1995] the manufacturer was not able to reproduce the overdose of Cox and Kidd cases. It was the user, i.e., the doctor and technician, who managed to reproduce it. The error forcing context in these cases is the operator completing changing the mode from x-ray to electron in less than 8 seconds. There were 11 units of THERAC-25 in use and the number of times the treatment was given successfully can be estimated. Therefore, the probability that the error forcing context occurs can be estimated.

---

<sup>3</sup> Race condition is an error condition in which two signals or sets of data collide. It can take place within a chip, a circuit, a network or an application, e.g., a software application. It can be due to a timing malfunction in the hardware or poorly written software.

In order to estimate the probability of the error forcing context occurrence, the information needs to be collected over a given time period. For each THERAC-25 unit, the total number of times of treatment and the number of times of failures caused by the error forcing context occurrence (the failures due to other reasons do not count) should be obtained from the operating records of each unit.

According to [Peterson 1995], 11 units of THERAC-25 were used between June 1985 and January 1987 and a total of 6 accidents/accidents occurred, with 1 patient in Kennestone Regional Oncology Center (Marietta, Georgia), 1 patient in Ontario Cancer Foundation (Ontario, Canada), 2 patients in Yakima Valley Memorial Hospital (Washington), and 2 patients in East Texas Cancer Center (Tyler, Texas). Exact number of patients who have had the Therac-25 treatment is unavailable. However, [Peterson 1995] mentioned that around 500 patients had the treatment in East Texas Cancer Center. If we assume that the same number of the patients in other centers used Therac-25, the total patients number is about 5,500. The probability of the error forcing context is thus approximately  $6/5,500=0.0011$ .

#### **C.8.3.1.5 Failure Categorization**

Failure modes:

- System failure mode:
  - Software runs with wrong results that are not evident
- Element failure mode:
  - Input element can not handle the input properly, a race condition exists which caused the accidents

Failure causes:

Internal Causes:

- Incomplete software requirement analysis
- No software documentation
- Poor user interface
- Test plan was not implemented or executed appropriately in Testing and Validation
- Inadequate accident management

EFC:

- When an operator first selected “x-ray” treatment by mistake and quickly corrected it to “electron” treatment in less than 8 seconds, a part of the software is not aware of the correction, and high energy electron is used instead of the lower dose rate specified by the operator.

Failure effects and consequences:

- 6 overdoses and 4 fatalities

#### **C.8.3.1.6 Dependent Failure and CCF**

No.

### **C.8.3.1.7 Discussion**

Leveson [Leveson 1993] stated “the description AECL provided for the FDA, although we have tried to clarify it somewhat. The description leaves some unanswered questions, but it is the best we can do with the information we have.” She probably did not perform a detailed failure analysis.

Leveson found that the claim that safety had increased  $10E+5$  times skeptical. A bad reliability claim by a vendor should not prevent good reliability/risk analysis.

### **C.8.3.2 London Ambulance Dispatch System [South 1993, Finkelstein 1996] - October 1992**

#### **C.8.3.2.1 Summary**

The London computer-aided dispatch system developed by the London Ambulance Service, which operated more than 700 ambulances and received between 2000 and 2500 calls daily, was put into service on October 26, 1992. This was the first time the system was fully implemented without being fully tested. The system allows the operators (call takers) to enter the calls, receives location and status information of ambulances through mobile data terminals and mobile radios, determines the nearest available ambulance, and allows the operators (allocators) to allocate the most suitable ambulance. Due to multiple factors, such as inadequate project management, incomplete software, and incomplete testing, (discussed in the next paragraph,) the system failed to dispatch ambulance timely, dispatched multiple ambulances to the same location, which resulted in many delays in the ambulance service and as many as 20 alleged deaths of the patients, although according to the coroner’s court in no case has the late arrival of an ambulance caused a patient's death. After the accident, the Central Ambulance Control reverted to a semi manual method of operation, identical to that which had operated with a variable degree of success before.

#### **C.8.3.2.2 Software Failures**

The software of London Ambulance Dispatch system is an application software which runs on Windows 3.0 and INTEL 486 processor in a multitasking environment. The software was developed by Systems Options Ltd, a small software company with no experience in similar applications. It relies on near perfect information of vehicle location and status, which are difficult to obtain in reality. Although some poor allocations may be attributable to errors in the allocation routine, it is believed that the majority of allocation errors were due the system not knowing the correct vehicle status and location. The poor location and status information was due to inadequate training of the operators and ambulance crew, radio communication black spots and bottle neck, and faults in the hand shaking routine between mobile data terminals and the dispatch system, etc. Incorrect status and position information resulted in incorrect allocations such as multiple vehicles sent to the same incident, or not the closest vehicle sent, and generation of many exception messages which moved off the screen which further delayed the response of the operators.

The system was over ambitious and developed against impossible timetable. The project management was inadequate and the decision to implement the system without fully tested was a mistake. The system was not complete, not properly tuned, and not fully tested. There were outstanding communication problems to and from the mobile data terminals. The users of the system, both the operators and ambulance crew, were not adequately trained.

#### **C.8.3.2.3 Consequence**

The deaths of 20 patients were suspected to be linked to the delays caused by the system. The chief executive of the London Ambulance System resigned.

#### **C.8.3.2.4 Likelihood of Error Forcing Context**

The use of the system is the error forcing context. The probability of EFC is 1.

#### **C.8.3.2.5 Failure Categorization**

Failure Modes:

- System failure mode:  
Runs with evidently wrong results
- Element failure mode:  
Data error of INPUT and OUTPUT elements (unknown status and location of vehicles, communication problems from and to mobile data terminal)  
Incorrect implementation of a function in PROCESSING element (some problems were due to errors in allocation routine)

Failure Causes:

Internal Causes:

- Incompatibility between software and overall system in System/information engineering and modeling stage (incompatibility between dispatch system and data acquisition system)
- Impact conditions not taken into account in Software requirement analysis stage (impacts of incomplete information on dispatch system, i.e., poor robustness of the dispatch system software)
- Incomplete test plan and/or test procedures in Test stage (system was not fully tested)
- Test plan was not implemented or executed appropriately in Test stage (system was not fully tested)

EFC:

Use of the ambulance dispatch system

Failure effects and consequences:

Deaths of 20 patients were suspected to be linked to the delays caused by the system

#### **C.8.3.2.6 Dependent Failure and CCF**

No.

#### **C.8.3.2.7 Discussion**

No.

### **C.8.3.3 China Airline Flight B1816 Crash at Nagoya [Ladkin, CSE] - 4/26/1994**

#### **C.8.3.3.1 Summary**

An A300 airplane B1816 operated by China Airline crashed at Japan's Nagoya Airport in April, 1994 during the landing process. Before the airplane was landing, it was under manual control by F/O (First Officer). The plane went into take-off/go-around because the F/O inadvertently activated the GO AROUND lever, which changed the FD (Flight Director) to Go Around mode. This made the aircraft deviate above its normal glide path. Under these conditions the F/O continued to push the control wheel despite its strong resistive force in accordance with the captain's instructions, i.e., the autopilot attempted to control the airplane in a way that was directly opposite to what the F/O was attempting to control. Obviously, the design of the autopilot software does not allow the pilot to disconnect it in case of this conflict. Thus, the trimmable horizontal stabilizer moved to its full nose-up position (this conflicted with the movement of elevator) and caused an abnormal out-of-trim situation. The crew were unaware of abnormal situation because there was no warning about this abnormal situation.

The angle of attack (AOA, the angle the wing makes relative to airflow over the wing) increased, Alpha Floor function to enter an optimal climb configuration was activated and the pitch angle increased. It is considered that the captain (who had now taken the controls) judged that the landing would be difficult and opted for go-around. The aircraft began to climb steeply with a high pitch angle attitude. The captain and the F/O did not carry out an effective recovery, and aircraft stalled and crashed, killing 264 of 271 people on board.

#### **C.8.3.3.2 Software Failures**

In addition to human errors involved in this accident, the most likely reason is not solely the software itself but the confused interactions between software and human pilot [CSE]. Two minutes before the landing, the autopilot went into take-off/go-around mode, which caused the airplane to continue climbing. It is thus impossible for the plane to land. On one hand, the autopilot was attempting to gain altitude and increase the pitch of the plane, while the F/O was trying to decrease the altitude using different parts of the plane. The crew had to switch the autopilot out of take-off/go-around mode but could not undo some of the changes the autopilot made to the stabilizer flaps on the wings that increased the altitude. The crew had to switch the autopilot into go-around mode again.

The main contributing factor to the accident is the autopilot software could not solve the conflicts between itself and human pilot. The interface of software design of the autopilot was not optimal for communication between pilot and autopilot, e.g., there were no audio cues signifying when the autopilot was engaged or disengaged. Also, below certain critical altitude, the autopilot was still activated because the software designer were afraid that there was insufficient time for a human pilot to regain the control in this situation.

Therefore, the error forcing context is the conflict between autopilot and human pilot, i.e., the pilot attempts to control the aircraft without knowing that the autopilot is engaged after the pilot inadvertently activating the GO AROUND lever.

#### **C.8.3.3.3 Consequences**

264 of 271 people on board were killed.

#### **C.8.3.3.4 Likelihood of Error Forcing Context**

The likelihood of error forcing context can be represented in terms of the probability that the F/O inadvertently activated the GO AROUND mode of autopilot, and failed to recognize it. It can be estimated by collecting actual data and performing human reliability analysis.

#### **C.8.3.3.5 Failure Categorization**

Failure modes:

– System failure modes:

Confusing or less informative interface

Runs with wrong results that are not evident

– Element failure modes:

Unintended function (allows both autopilot and pilot controls) in PROCESSING element

Omission of a function (alarm or warning messages to make pilot aware of situation) in OUTPUT element

Failure causes:

Internal Causes:

Desired functions (autopilot should warn the situation) are not specified in the requirements in Software requirement analysis stage

Incorrect specification (conflict controls from both autopilot and human pilot are allowed) in Software requirement analysis stage

EFC:

The F/O inadvertently activated the GO AROUND mode, and failed to recognize it.



Failure effects and consequences:

264 of 271 people on board were killed.

#### **C.8.3.3.6 Dependent Failure or CCF**

No.

#### **C.8.3.3.7 Discussion**

On June 8, 1994, BEA (Bureau Enquetes Accidents) transmitted the following recommendation to DGAC (Direction General de l'Aviation Civile), France: "We recommended that study be performed for the modification of the aircraft, with all necessary accompanying measures, leading to the connection of autopilot when a pilot overrides it while in Landing and Go Around modes. The modification resulting from this study should be made mandatory."

#### **C.8.3.4 Turkey Point Diesel Generator Sequencer - November 4, 1994**

##### **C.8.3.4.1 Summary**

Turkey Point Units 3 and 4 have four sequencers (one sequencer per train, per unit). Each sequencer is provided with Manual test and Automatic Self-test capability. The test mode is determined by a three-position Test Selector switch. The three positions are AUTO (self-tests 16 steps or scenarios in the automatic test sequence), MAN (each test is manually initiated), and OFF (no test signals are generated). With the sequencer Test Selector switch in AUTO, the sequencer steps sequentially through sixteen steps; first five bus stripping/clearing steps, followed by eleven LOOP and/or LOCA scenarios. On November 3, 1994, Turkey Point Unit 3 was operating in Mode 1 at 100% power, and Unit 4 was in Mode 5 during a refueling outage. During the Unit 4 Integrated Safeguards Test, a failure of the 3A sequencer to respond to the opposite unit's Safety Injection (SI) signal occurred. The 3A sequencer response should have been to start the 3A High Head Safety Injection (HHSI) pump. However, the pump failed to start because it did not receive a start signal from the sequencer. Troubleshooting resulted in the discovery of a defect in the sequencer software logic which, under certain conditions, could inhibit the sequencer from responding to a valid emergency signal. The defect manifested itself in the failure of the 3A HHSI pump to start. This event is documented in LER 250-1994-005-02.

The software logic defect is limited to the test function, but the defect is common to all four sequencers. Since this condition is applicable to both the automatic self-test and manual testing, the sequencers must be considered inoperable during both testing modes. The design intent of the sequencers is such that should a "real" emergency signal occur while the sequencer is being tested, the test signal clears, allowing actuation of the Engineered Safety Features controlled by the sequencer.

On May 5, 1994, there was an inadvertent ESF actuation on Unit 3, in which all equipment responded as design, except the 4A HHSI pump. LER 250-1994-002 documents this other event. At that time the failure of the 4A HHSI pump was attributed to an intermittent failure, which could not be reproduced. As a result of the discovery of the software logic defect that is active during the test function, the licensee was able to reproduce it at will on the sequencer simulator. The licensee believes that the 4A HHSI pump failed to start because of the same defect that caused the 3A HHSI pump failure to start, i.e., the software logic defect that is active during the test function.

The detailed review of the sequencer software resulted in the discovery of one other error in the software, which is independent of the test mode; a potential condition was identified which would preclude the automatic start of the Containment Spray (CS) pumps. The condition identified occurs when the Hi-Hi Containment Pressure (HHCP) signal is received by the sequencer during an approximate 60 millisecond (ms) time window just prior to the end of sequencer load block 3 for Loss of Coolant Accident (LOCA) or Loss of Offsite Power coincident with LOCA (LOOP/LOCA) events. This event also is documented in LER 250-1994-005-02.

#### **C.8.3.4.2 Software Failures**

A software design defect was discovered whereby the start signal for the 3A HHSI pump remained inhibited during both manual and automatic testing, even though a valid process input was present. This software logic defect was introduced during the detailed logic design phase of the software development. The detailed logic designer and the independent verifier failed to recognize the interaction between some process logic inhibits and the test logic. The defect in the software logic was not detected during the Validation and Verification process (V&V) because the response to valid inputs was not tested during all stripping and loading sequences of the automatic and manual testing logic.

An "Independent Assessment Team" (IAT) confirmed that the V&V was not comprehensive enough to test certain aspects of the logic: "The plan was weak in that it relied almost completely on testing as the V&V methodology. More emphasis on the analysis of the requirements and design would have increased the likelihood of discovering the design flaw."

The cause of the error in the sequencer software that would preclude the automatic start of the Containment Spray (CS) pumps was not found in the LER. Possibly, the cause is the same as the one for the other software error, i.e., a software logic defect introduced during the detailed logic design phase of the software development.

#### **C.8.3.4.3 Consequence**

The periodic inoperability of all four sequencers, as described above, has existed since the sequencers were installed during the dual unit outage in 1990/1991. As a result of the erroneous inhibit signals, the potential exists for any sequencer output to be prevented from being generated when required. Exactly which output or outputs is (are) prevented is determined by a combination of factors, i.e.,

which test scenario is in progress, how long since the test scenario was initiated, and which process input or inputs are received.

Depending on the type of SI signal (e.g., LOCA on the same train, or LOCA on other Unit), the software defect would cause several components to not be automatically loaded by a sequencer. For example, if there is a LOCA on the same train, the following equipment would not be automatically loaded by the 3A sequencer: Residual Heat Removal Pump 3A, HHSI Pump 3A Intake Cooling Water Pumps 3A (1) and 3C (1), Emergency Containment Cooler Fan 3B and 3C, Component Cooling Water Pumps 3A (1) and 3C (1), Emergency Containment Filter Fans 3B and 3C. The equipment lists would be similar for the other three sequencers. The equipment identified with “(1)” may already be in operation and may not require manual action to start.

In general, for the approximate one-hour duration of each test step (with the Test Selector switch in AUTO), the sequencer will not respond correctly to a valid process input signal. The defect in the sequencer test logic represented a potential concern for events where SI is required for mitigation and no LOOP is experienced. Because the sequencers would not have responded properly to an SI signal as designed, Turkey Point Units 3 and 4 were operating outside their design basis.

The LER considered the failure of the automatic start of the Containment Spray (CS) pumps to be not significant, in part because the manual start capability of the CS pump is not affected (and is adequately proceduralized), and in part because the probability of occurrence of the condition is lower than the probability of failure of both trains of containment spray (see below).

#### **C.8.3.4.4 Likelihood of Error Forcing Context**

Separate errors in the sequencer software caused 1) failure of a sequencer to respond to an SI signal, and 2) failure of a sequencer to automatically start the CS pumps. A different EFC is associated with each error, and is discussed separately.

##### **1) Error in the sequencer software causing failure of a sequencer to respond to an SI signal**

This logic defect can occur when the sequencer is in either the manual or automatic test mode, and the test sequence currently being executed is loading sequence test 2, 3, 6, 8, or 10. These five test steps are all in the loading sequence test steps, so the first affected step is the seventh step in the total testing sequence. During each of these affected test steps, fifteen seconds after the initiation of the step, the sequencer would not have responded properly to a valid process input signal. Thus, in general, for the approximate one-hour duration of each of the above test steps (with the Test Selector switch in AUTO), the sequencer will not respond correctly to a valid process input signal. Hence, the sequencer was inoperable for about five hours out of each sixteen hour period as long as its Test Selector switch was in AUTO. The sequencer was also inoperable for the duration of any Manual test of the five test steps listed above. A complete manual test on one sequencer takes about one hour. Hence, in general, the EFC is the sequencer executing the test associated with each of these five test steps.

The review of the sequencer logic determined that improper operation of the sequencer could occur for only certain sequencer stripping/loading scenarios in which an SI signal without LOOP occurs. The licensee identified the following four potential plant events where the logic software defect could affect the operation of the sequencer, depending upon which of the five affected test steps are being performed when the SI signal is received by the sequencer:

- #1 LOCA Same Train
- #2 LOCA on other Unit
- #3 LOCA with High High Containment Pressure (HHCP) < 13 seconds
- #4 LOCA with HHCP > 13 seconds.

For each of these events, the sequencer could receive a valid SI signal but the logic defect could inhibit the sequencer from starting equipment. Events #1, #3, and #4 above each have four logic test steps out of a total of sixteen which would inhibit the sequencer from providing a start signal to the equipment it controls, while event #2 is affected by only one of the sixteen logic test steps.

The probability that an individual sequencer would not respond to a valid same train SI signal is 4 hours/16 hours = 2.5E-1. The probability that an individual sequencer would not respond to a valid opposite unit SI signal is 1 hour/16 hours = 6.25E-2. These probabilities are conditional on an individual sequencer being in a testing mode, and the occurrence of one of the four LOCAs identified above.

## 2) Error in the sequencer software causing failure of automatic start of the CS pumps

The EFC for the failure of the software to automatically start the CS pumps is a HHCP signal received by the sequencer during an approximate 60 ms time window just prior to the end of sequencer load block 3 for LOCA or LOOP/LOCA events. Hence, the failure to automatically start a CS pump due to this software error can only occur during an approximate 60 ms time window. If it is assumed that HHCP can occur at any time within approximately two minutes after the SI signal (the earliest time at which SI is postulated to be reset), then the probability of the evaluated condition occurring on one train is:

$$0.060 \text{ sec}/(2 \text{ min} \times 60 \text{ sec/min}) = 5.0\text{E-}4$$

This probability is conditional on the occurrence of a signal of HHCP.

Using the Turkey Point baseline Probabilistic Safety Assessment (PSA) model, the probability of dual train failure of the CS system if called on to operate was estimated to be approximately 2.6E-3. This estimate reflects CS system and support system component failure probabilities not including either of the software errors reported here. The estimate of the probability of a CS pump not starting automatically in a LOCA or LOOP/LOCA due to the reported software error is therefore approximately a factor of five below the PSA's estimated probability of failure of both CS trains.

According to the LER, the probability of the software error affecting both trains is considerably lower, since it would require: 1) the initiating SI signals to be at the sequencer inputs within 60 ms of each other; 2) the two signals of HHCP both occurring within the 60 ms window of vulnerability; 3) the sequencer input processing times to be identical; and 4) the timing of the two sequencers in synchronization. The difference in the cumulative delay time for relay actuations on the two trains of Engineered Safety Features Actuation System (ESFAS) and differences in sequencer processing would likely be sufficient to preclude the condition on both trains.

#### **C.8.3.4.5 Failure Categorization**

Failure modes:

- System failure mode:
  - Runs with wrong results that may not be evident.
- Element failure mode:
  - One of the elements of the software (possibly, the processing element) incorrectly implemented some functions of the sequencer.

Failure causes:

Internal causes:

According to the LER, the software error causing failure of a sequencer to respond to an SI signal was introduced during the detailed logic design phase of the software development. Hence, the error was introduced during the stage “System analysis and design” of the software development. The cause of the error in the sequencer software that would preclude the automatic start of the CS pumps was not found in the LER. Possibly, the cause is the same as the one for the other software error.

EFC:

- Regarding the error in the sequencer software causing failure of a sequencer to respond to an SI signal, in general, the EFC is the sequencer executing the test associated with sequence tests 2, 3, 6, 8, or 10.
- Regarding the error in the sequencer software causing failure of a sequencer to automatically start the CS pumps, the EFC is a HHCP signal received by the sequencer during an approximate 60 ms time window just prior to the end of sequencer load block 3 for LOCA or LOOP/LOCA events.

Failure effects and consequences:

- The periodic inoperability of all four sequencers, as described above, has existed since the sequencers were installed during the dual unit outage in 1990/1991. As a result of the erroneous inhibit signals, the potential exists for any sequencer output to be prevented from being generated when required. Because the sequencers would not have responded properly to an SI signal as designed, Turkey Point Units 3 and 4 were operating outside their design basis. The LER considered the failure of the automatic start of the Containment Spray (CS) pumps to be not significant.

#### **C.8.3.4.6 Dependent Failure and CCF**

Turkey Point Units 3 and 4 have four HHSI pumps; one per train, per unit. Each HHSI pump is capable of providing 50 percent of system requirements, therefore two of the four are required to mitigate the consequences of accidents analyzed in the Updated Final Safety Analysis Report (UFSAR). To meet single failure criteria, each sequencer signals its associated HHSI pump to start, and the opposite unit's sequencers signal their associated HHSI pumps to start. For example, an SI signal on Unit 3, Train A, signals the 3A sequencer and both of the Unit 4 sequencers. With no equipment failures, all four HHSI pumps will respond to an SI signal on either unit. The software logic defect is limited to the test function, but the defect is common to all four sequencers (one sequencer per train, per unit). Hence, the error in the sequencer software causing failure to respond to an SI signal can be considered a common cause failure of the four sequencers.

The error in the sequencer software causing failure to automatically start the CS pumps also can be considered a common cause failure of the four sequencers.

#### **C.8.3.4.7 Discussion**

As discussed above, this event illustrates:

- 1) the potential of a software error to fail redundant channels of a system when the channels use the same software.
- 2) the potential of a software error to cause a nuclear plant to operate outside its design basis during several years.

#### **C.8.3.5 Common Cause Failure of Voltage Regulating Transformers and Vital AC Buses at Pilgrim - 4/1/1997**

##### **C.8.3.5.1 Summary**

On April 1, 1997, Pilgrim nuclear power station was in cold shut down, with the reactor mode selector switch in the REFUEL position. During a severe storm (blizzard), the safety-related 120 volt (alternating current) safeguards control power Bus 'A' panels Y3 and Y31, and Bus 'B' panels Y4 and Y41 de-energized on two occasions while the 4.16 Kv distribution system including safety-related Buses A5 and A6 and related electrical system were energized from the 345 Kv transmission system. Panels Y3/31 are powered from Bus A5 via 480/120 volt regulating transformer X55. Similarly, panels Y4/41 are powered from Bus A6 via 480/120 volt regulating transformer X56. The 120 volt safeguards buses were de-energized after brief, severe 345 Kv transmission system undervoltage transients that resulted in automatic shut downs of the voltage regulating transformers X55 and X56 that power these buses. Subsequently, a loss of preferred off-site power (345 Kv) followed later by a loss of secondary off-site power (23 Kv) occurred while the emergency diesel generators were in operation. These losses of the off-site power sources were caused by the effects of the storm.

This event is documented in LER 293-1997-007.

### **C.8.3.5.2 Software Failures**

The cause of the de-energizing of panels Y3/31 and Y4/41 on April 1, 1997, at 0135 hours and 0257 hours, was the automatic shut downs of voltage regulating transformers X55 and X56. During the time frames of the shut downs of the regulating transformers, the 345 Kv transmission system experienced brief but severe voltage transients, to as low as 250 Kv. The corresponding voltage on the 480 volt load center portion of the 4.16 Kv distribution system was as low as 350 volts and lasted for approximately 6 - 8 cycles (0.130 seconds). Regulating transformers X55 and X56 were designed and tested to reliably regulate input voltages of 480 volts 20 percent (384 - 576 volts) and provide regulated output voltages of 120 volts  $\pm$ 4 percent. Each regulating transformer contains a programmable microprocessor control unit (MCU). The MCU is a 40-pin integrated circuit chip that senses input voltage and selects the proper voltage tap to provide the regulated 120 volt output voltage. The software code contained in an MCU automatically shut down its regulating transformer if input voltage was outside the input voltage range of 480 volt  $\pm$ 20 percent (384 to 576 volts).

The automatic shut downs of the regulating transformers occurred due to a deficiency in the licensee's specification of the transformers because it did not address the effects of 480 volt transients of less than 384 volts (greater than zero volts). The cause of the deficiency was, apparently, a cognitive (unintentional) error made by the utility electrical engineer who prepared the specification. Hence, the cause of the software failure is inadequate specification of requirements of the software. A contributing cause was the manufacturer and supplier documentation that did not identify the automatic shut down feature of the transformers due to voltage transients of less than 384 volts (greater than zero volts). The lack of the identification of the feature is significant because an automatic shut down due to input voltages greater than zero volts but less than 384 volts would require a manual reset of the transformer, versus a designed automatic reset if input voltage was zero volts.

### **C.8.3.5.3 Consequence**

Circuits powered from panels Y3/31 and Y4/41 that were affected include:

- 1) Normally energized logic relays that are part of the inboard and outboard circuitry of the primary containment isolation control system (PCIS) and reactor building isolation control system (RBIS).
- 2) Torus temperature and pressure monitoring systems 'A' and 'B'.
- 3) Hydrogen/oxygen monitoring systems 'A' and 'B'.
- 4) Post accident monitoring systems 'A' and 'B'.
- 5) Instrument and control system 'A' and 'B'.

- 6) Noble gas effluent radiation monitors 1001-608, 1001-609, and 1001-610.
- 7) Containment atmosphere control system 'A' and 'B'.
- 8) Anticipated transient without scram (ATWS) division 1 and 2 alternate current power supplies. The system is also equipped with redundant power supplies that are powered by 125 vdc power.
- 9) Analog trip system 'A' and 'B' alternate current power supplies. The system is also equipped with redundant power supplies that are powered by 125 vdc power.
- 10) Salt service water (SSW) system train 'A' and 'B' pressure switches and reactor building closed cooling water (RBCCW) system train 'A' and 'B' pressure switches. The pressure switches monitor SSW and RBCCW header pressures, and provide the automatic start signal(s) to the systems' pumps if sufficient header pressure is not present after applicable time delay(s).

The de-energizing of panels Y3 and Y4 resulted in:

- 1) A PCIS Group 6 isolation signal and resultant automatic closing of the reactor water cleanup (RWCU) system isolation valves MO-1201-2, MO-1201-5, MO-1201-80, trip of the RWCU pump that was in service, and interruption in RWCU system operation.
- 2) An RBIS isolation signal and resultant automatic start of the standby gas treatment system (SGTS) trains 'A' and 'B' and automatic closing of the reactor building ventilation supply and exhaust dampers.

Regulating transformer X58 is part of the power supply for the post accident sampling system train 'B' equipment and also shut down when X55 and X56 shut down. Regulating transformer X57 is part of the power supply for the post accident sampling system train 'A' equipment and was tagged out of service for maintenance when the storm occurred.

The salt service water SSW pump P-208D and the RBCCW pumps in trains 'A' and 'B' that were in service at the time of the event stopped, not as a result of the de-energizing of panels Y3/31 and Y4/41, but due to brief undervoltage transients on the 480 volt portion of the 4.16 Kv distribution system while safety-related Buses A5 and A6 were powered from the 345 Kv transmission system. During the period of time panels Y3 and Y4 are de-energized, the SSW trains 'A' and 'B' pumps and swing pump 'C', and the RBCCW trains 'A' and 'B' pumps would not be capable of automatically starting as assumed in the design. The manual start function of the pumps is not affected while the respective panel is de-energized.

Panels Y3/31 and Y4/41 de-energized twice; in one occasion for a maximum of approximately 24 minutes. The LER concluded the loss of power to panels Y3/31 and/or Y4/41 is detectable, and the actions to re-energize the panels are proceduralized.



SSW pump P-208D and one pump each RBCCW train were manually started via their control switches about 20 - 30 seconds after the pumps stopped. Regulating transformers X55 and X56 were reset in accordance with procedures, and panels Y3/31 and Y4/41 also were re-energized. After the PCIS Group 6 circuitry was reset, the RWCU system was returned to service. After the RBIS circuitry was reset, the SGTs was returned to standby service, and the reactor building ventilation system was returned to normal service.

The undervoltage shut downs of the regulating transformers, although in accordance with the software code contained in the respective transformer microprocessor control unit, was outside the Pilgrim Station design basis.

The microprocessor control units (MCUs) for regulating transformers X55, X56, X57, and X58 were modified during the week of April 6, 1997, to disable the undervoltage and overvoltage shut down functions. After this change, the transformers operate in the unregulated mode when the input voltage is outside the design range.

#### **C.8.3.5.4 Likelihood of Error Forcing Context**

The software code contained in an MCU automatically shut down its regulating transformer if input voltage was outside the input voltage range of 480 volt  $\pm$ 20 percent (384 to 576 volts). Hence, the EFC was an event, such as the severe storm, that could cause the 480 volt load center portion of the 4.16 Kv distribution system to be below 384 volts. Hence, the likelihood of the EFC is the probability of such event. According to NUREG/CR-5496, "Evaluation of Loss of Offsite Power Events at Nuclear Power Plant: 1980-1996," the frequency of loss of offsite power (LOOP) due to plant-centered (during power operation), grid-related, and severe-weather causes is about 6E-2 per year. Assuming that an event that causes a LOOP will also cause the 480 volt load center to have significantly low voltages, the probability that the EFC occurs in any given year is about 6E-2.

#### **C.8.3.5.5 Failure Categorization**

Failure modes:

- System failure mode:

- Runs with evidently wrong results.

- Element failure mode:

- One of the elements of the software (possibly, the processing element) of an MCU has the unintended function of shutting down the regulating transformer when the input voltage is less than 384 volts (greater than zero volts).

Failure causes:

- Internal causes:

- Inadequate requirements of the software, in particular, unspecified exception conditions.

EFC:

- The software code contained in an MCU automatically shut down its regulating transformer if input voltage was outside the input voltage range of 480 volt  $\pm 20$  percent (384 to 576 volts). Hence, the EFC was an event, such as the severe storm, that could cause the 480 volt load center portion of the 4.16 Kv distribution system to be below 384 volts.

Failure effects and consequences:

- The undervoltage shut downs of the regulating transformers, although in accordance with the software code contained in the respective transformer microprocessor control unit, was outside the Pilgrim Station design basis.

#### **C.8.3.5.6 Dependent Failure and CCF**

The MCUs use the same software, and the failure of the software caused both MCUs and regulating transformers to fail. Hence, a common cause failure of the regulating transformers X55 and X56 occurred due to failure of the software of the MCUs.

#### **C.8.3.5.7 Discussion**

This event illustrates the potential of a software failure to propagate into the failure of several redundant trains of equipment when these trains use the same software.

### **C.8.3.6 Core Protection Calculators Inoperable at Palo Verde 2**

#### **C.8.3.6.1 Summary**

The Core Protection Calculators (CPCs) consist of four separate, redundant channels. Each channel is a computer system that continuously calculates thermal conditions and thermal limits. The CPC system is an integral part of the plant protective system in that it provides two trips to the reactor protection system (RPS): Departure from Nucleate Boiling Ratio (DNBR) and Local Power Density (LPD). Trip signals are provided to the RPS whenever the minimum departure from nucleate boiling ratio (DNBR) or fuel design limit Local Power Density is approached during reactor operation. Each CPC channel provides contact outputs to its respective RPS channel. The CPC system is a four channel system that uses a two out of four logic for reactor trip signal generation.

The following analog input sensors are processed in each CPC channel:

- 2 Cold Leg Temperatures
- 2 Hot Leg Temperatures
- 1 Pressurizer Pressure
- 3 Ex-core Neutron Flux Detectors

In the event of a failure of one of the input sensors a trip signal for the applicable CPC channel should be generated.

Each input parameter is read by two separate analog input modules in a channel. One of the two redundant analog input modules is normally selected. In the event the normally selected module indicates a failure, the software will select the alternative module. In the event of a failure of both modules at the same time, the CPC uses the last known good value and a trip signal for that channel should be generated.

Detectable CPC channel failures, resulting in a loss of protective function and channel inoperability, are required to generate CPC Fail indication and associated Low DNBR and High LPD channel trips. Input failures resulting in a sensor out of range affecting one or more CPC process inputs will result in a CPC Sensor Failure indication. In addition, since the CPC software limits the sensor value to the lower or upper range limit value, a CPC channel trip would be generated in most cases due to these extreme values.

Software release 6.1 was installed into the Unit 2 CPCs in May 2005. On May 18, 2005, personnel of Westinghouse (the vendor of this release) identified a problem with the installed version of the CPC software for Unit 2. On August 8, 2005 Westinghouse personnel completed an apparent cause analysis for the issue and concluded the issue was a nuclear safety concern. At 0900 hours on August 22, 2005, a Westinghouse engineer informed the Palo Verde staff of the issue with the CPC software. At 1326 hours on August 22, 2005, Unit 2 was operating in Mode 1, Power Operation, at approximately 100% power when control room personnel declared all four channels of the CPCs inoperable, and the licensee made the decision to enter Technical Specification Limiting Condition for Operation (LCO) 3.0.3 due to the installed CPC software not supporting Technical Specification Bases 3.3.1. Plant shutdown commenced at 1605 on 8/22/05 and LCO 3.0.3 was exited at 1750 when the unit entered Mode 3, Hot Standby.

This event is documented in LER 529-2005-004.

#### **C.8.3.6.2 Software Failures**

Personnel of Westinghouse (the vendor) discovered that release 6.1 of the Unit 2 CPC software was not consistent with the system requirements regarding the system response to analog input module errors. When both analog input modules within a CPC channel indicate an error simultaneously, the CPC uses the last known good value. However, the system requirements state that a channel trip should be initiated for this event. Software release 6.1 resulted in the CPCs not being able to generate this trip signal.

#### **C.8.3.6.3 Consequence**

The installed version (release 6.1) of the Unit 2 CPC software was not consistent with the system requirements regarding the system response to analog input module errors since the software was installed in May 2005. Hence, it appears that all four channels of the CPCs were inoperable, and the

plant operation violated Technical Specifications since that date. In addition, the plant had to be shutdown. However, both a sensor failure and an analog input module failure actuate contact output signals in the affected channel to the CPC Operator's Module Alarm and the plant annunciator alarm in the main control room which would alert the control room operators to the condition. The LER points out that the event did not result in a transient more severe than those analyzed in the updated Final Safety Evaluation Report Chapters 6 and 15, and concludes that the event did not have significant negative safety consequences.

#### **C.8.3.6.4 Likelihood of Error Forcing Context**

A latent fault was in the installed software (release 6.1) of the Unit 2 CPCs, but it was discovered before it was triggered into an actual failure. The fault is that when both analog input modules within a CPC channel indicate an error simultaneously, a channel trip should be initiated, but the CPC was not able to generate this trip signal. Hence, the EFC is the simultaneous failure of both analog input modules within a CPC channel. Possibly, the EFC also includes failures of the analog sensors providing input to both analog input modules within a CPC channel.

Lacking probabilistic information about the analog input modules and about the analog sensors providing input to these modules, the probability of failure of both analog input modules within a CPC channel and of the associated analog sensors is not known. Hence, the likelihood of the EFC is not known at this time.

The software fault was common to all four CPCs, so there is a potential for common cause failure (CCF) of several or all CPCs. It appears that the most likely way in which this CCF may occur is some kind of dependent failure of several analog input modules or several of the associated analog sensors, or a combination of these two type of failures.

#### **C.8.3.6.5 Failure Categorization**

Failure modes:

- System failure mode:

  - Runs with potentially wrong results that are not evident.

- Element failure mode:

  - The software of the CPCs was not consistent with the system requirements regarding the system response to analog input module errors. If both analog input modules within a CPC channel fail, a channel trip signal should be generated, but it would not have been generated. Hence, there is an omission of the function that should generate this signal. One of the elements of the software (possibly, the processing element) was missing this function.

Failure causes:

- Internal causes:

  - The LER states that investigation into the cause of this event is ongoing, and that preliminary results indicate the direct cause is that a CPC system requirement

specification was not properly translated into the CPC software by the vendor. Accordingly, it appears that the error was introduced during the development of the software, possibly during the stage of “System analysis and design.”

EFC:

- The EFC is the simultaneous failure of both analog input modules within a CPC channel. Possibly, the EFC also includes failures of the analog sensors providing input to both analog input modules within a CPC channel.

Failure effects and consequences:

- All four channels of the CPCs were inoperable, and the plant operation violated Technical Specifications since the software was installed in May 2005. In addition, the plant had to be shutdown from approximately 100% power.

#### **C.8.3.6.6 Dependent Failure and CCF**

There was a potential for common cause failure because all four channels of the CPCs were affected by the software failure.

#### **C.8.3.6.7 Discussion**

This event illustrates the potential of a software failure to propagate into the failure of several redundant channels of equipment when these channels use the same software.

The CPCs in Unit 2 were upgraded in November 2003. As part of the corrective actions, on August 25, 2005 activities were completed to install CPC software version 6.3 in all four channels of Unit 2 CPCs.

Unit 1 was scheduled to receive the upgraded CPC system in the refueling outage that was in progress, and Unit 3 is scheduled to receive the upgraded CPC system in a future refueling outage. The problem identified in the LER is therefore limited to Unit 2. The LER points out that changes will be made to Units 1 and 3 upgraded CPCs, prior to their installation, to correct the problem.

#### **C.8.3.7 Slammer Virus in Davis-Besse Nuclear Power Plant [Schulin, Poulsen] - 1/25/2003**

##### **C.8.3.7.1 Summary**

Davis-Besse nuclear power plant was operated by FirstEnergy, the Ohio utility company that was the focus of the investigation of 2003 blackout. In January 25, 2003, the Slammer worm penetrated a computer network at Davis-Besse power plant [Schulin]. The infection of the Slammer worm generated a large amounts of data that caused many computers to cease from communication with other computers on the network [Poulsen]. As a result, the infection of the worm disabled a safety monitoring system and the plant process computer, which the plant personnel believed to be protected by a firewall, for nearly five hours.

### **C.8.3.7.2 Software Failures**

The incident was caused by virus Slammer worm from an external consultant who had a link to the plant's intranet and provided the application software running on MS-SQL server of the plant. The worm entered the plant network through the interconnected link by bypassing (it was connected behind the fire wall) the firewall of Davis-Besse power plant. Several routes into its computer network completely bypassed the security firewall and one of these routes allowed the Slammer worm to take up residence in the plant's Safety Parameter Display System (SPDS) that is used by the operators to monitor the status of the safety-related processes and conditions [Poulsen].

It was discovered the security patch that removed the vulnerability target by the MS-SQL server worm was not installed on the server. The security patch was released on July 10, 2002 [Poulsen].

The error forcing context in this case is thus 1) the plant's network has a connection behind the firewall, 2) the consultant's computer was infected by the virus and was used in the connection, and 3) the MS-SQL server did not install the security patch.

### **C.8.3.7.3 Consequences**

The infection of the worm caused many computers to cease from communication with other computers and disabled a safety monitoring system and the plant process computer for nearly five hours.

### **C.8.3.7.4 Likelihood of Error Forcing Context**

The likelihood of the EFC can be expressed in terms of the probabilities of the events defining the EFC.

### **C.8.3.7.5 Failure Categorization**

Failure modes:

- System failure modes:
  - Runs with evidently wrong results
- Element failure modes:
  - Data error in INPUT and PROCESSING elements (Slammer worm generated a large amounts of data that caused many computers to cease from communication with other computers on the network)

Failure causes:

- Internal cause:
  - Maintenance (security patch was not installed)
- External cause:
  - Cyber security (virus)

EFC:

Infection of Slammer worm of a computer that is interconnected to internal network of the plant with MS-SQL server without installing the security patch

Failure effects and consequences:

Insignificant because of the control and protection functions of Davis-Besse were not affected and Davis-Besse was off-line at that time. However, the infection of the worm caused many computers to cease from communication with other computers and disabled a safety monitoring system and the plant process computer for nearly five hours.

#### **C.8.3.7.6 Dependent Failure or CCF**

The spread of the Slammer virus is a dependent failure. It has the potential to further propagate from the plant network to the corporate network. [Poulsen] also quoted an occurrence at one utility, in which the Slammer worm downed the critical Supervisory Control and Data Acquisition (SCADA) system after moving from a corporate network, through a remote computer to a VPN connection to the SCADA.

#### **C.8.3.7.7 Discussion**

NRC advised against the interconnection of a nuclear plant's network to an outside network [NRC 2003]. NRC also urged vendors to add additional security to their software development process, as a bulwark against sabotages writing backdoor into code, or implanting logic bombs programmed to shut down a safety at a particular time.

### **C.8.3.8 Natural Gas Pipeline Explosion in Soviet Union [Reed 2004, Detroit 2004, Loney 2004] - Summer 1982**

#### **C.8.3.8.1 Summary**

The economy of the Soviet heavily depended on production and transportation of oil and natural gas. In the early nineteen eighties, a new trans-Siberian pipeline was built to deliver natural gas from Urengoi gas fields across Kazakhstan, Russia, and Eastern Europe, into the West in order to make hard currency [Reed 2004]. Sophisticated control systems were needed to automatically operate pipeline. Computer hardware was bought in the open market but the attempt to buy software was turned down by the U.S. The Soviet had to look somewhere else. A KGB operative was sent to penetrate a Canadian software supplier in an attempt to steal the needed codes.

At that time, the United States was attempting to block Western Europe from importing Soviet natural gas. There were also signs that the Soviets were trying to steal a wide variety of Western technology. The CIA responded by modifying the software that the Soviet needed. This was the software that

later triggered a huge explosion in a Siberian natural gas pipeline. U.S. satellites picked up the explosion that occurred in the summer of 1982.

#### **C.8.3.8.2 Software Failures**

As mentioned above, there were signs that the Soviets were trying to steal a wide variety of Western technology. A KGB insider revealed the specific "shopping list" and the CIA slipped the flawed software to the Soviets. The pipeline software that ran the pumps, turbines, and valves was programmed to go haywire, after a decent interval, to reset pump speeds and valve settings to produce pressure far beyond those acceptable to the pipeline joints and welds. The faulty software, which was in a shopping list of Soviet priorities focusing on stealing Western technology [Loney 2004], was slipped to Russia.

Thus, the error forcing context is thus the installation of the software without identifying the flaw for the period of time before the explosion.

#### **C.8.3.8.3 Consequences**

The flawed software caused the biggest non-nuclear explosion the world has ever seen. It is estimated that there were no fatalities in the explosion but the damage to the Soviet economy was significant.

#### **C.8.3.8.4 Likelihood of Error Forcing Context**

The likelihood of the error forcing context can be represented in terms of the probability of 1.0 after the installation.

#### **C.8.3.8.5 Failure Categorization**

Failure modes:

- System failure mode:  
Runs with wrong results that are not evident
- Element failure mode:  
Incorrect implementation of unknown elements in the pipeline software

Failure causes:

- External cause:  
Cyber security

EFC:

After certain time interval after the faulty software installation without identifying the flaw



Failure effects and consequences:

The flawed software caused the biggest non-nuclear explosion the world has ever seen. According to the author of the book “At the Abyss” [Reed 2004], there was no evidence of fatalities in the explosion but the damage to the Soviet economy was significant.

#### **C.8.3.8.6 Dependent Failure or CCF**

No.

#### **C.8.3.8.7 Discussion**

No.

### **C.8.3.9 Maroochy Water Treatment Plant Accident [Age 2003, Red 2003, Datz 2004] - 2000**

#### **C.8.3.9.1 Summary**

From early 2000, a number of faults and communication breakdowns occurred to a network of 150 computer-controlled sewage pumping stations belonging to Queensland's Maroochy Shire Council in Australia. The pumps, alarms, and communication were not working as expected. Over the 2 month period, total 46 incidents occurred and this caused as much as 1 million liters of sewage to spill into parks and rivers. An engineer who quit his job from a company that installed Maroochy's computerized sewage control system did this using a laptop and a radio transmitter from his car to release the valves of the sewage control system.

#### **C.8.3.9.2 Software Failures**

The control systems of the water treatment plant were attacked maliciously by personnel who is familiar with the computer control system of the water treatment plant. Since these control systems are old, there is no security available to track or prevent this from happening. The engineer was caught because he parked in the wrong place and police recognized the computer and the radio equipments as having recently been stolen.

Thus, the error forcing context is when the water treatment plant is attacked by a person who is either an insider or knowledgeable enough (e.g., the engineer who quit his job from the company that installed the sewage control system) to attack the plant's control system remotely.

#### **C.8.3.9.3 Consequences**

Leakage of around 1 million sewage into the rivers and parks and \$50,000 to clean it [Datz 2004].

#### **C.8.3.9.4 Likelihood of Error Forcing Context**

The computer systems of the water treatment plant do not have security systems. Thus, the likelihood of the error forcing context is probability of an attack by a person who knows the control system of the plant.

#### **C.8.3.9.5 Failure Categorization**

Failure modes:

- System failure mode:  
Runs with evidently wrong results
- Element failure mode:  
Omission of a function (security) in COMMUNICATION and/or OUTPUT element

Failure causes:

Internal cause:

Missing functions (desired functions such as security are not specified in the requirements in Software requirement analysis stage)

External cause:

Cyber security (control from software was taken over by an intruder)

EFC:

Error forcing context is when the water treatment plant is attacked by a person who is either an insider or knowledgeable enough to attack the plant's control system remotely.

Failure effects and consequences:

Leakage of around 1 million sewage into the rivers and parks and \$50,000 to clean.

#### **C.8.3.9.6 Dependent Failure or CCF**

No.

#### **C.8.3.9.7 Discussion**

This case has been studied by the US National Infrastructure Protection Center and it has been used at international and national security conferences as the only known example of someone successfully taking over a data-control system to attach public infrastructure [Red 2003].

## **C.8.3.10 Blackout of North America [US 2004, Jesdanun 2004] - August 14, 2003**

### **C.8.3.10.1 Summary**

On August 14, 2003, large portions of the Midwest and Northeast United States and Ontario, Canada, experienced an electric power blackout. The blackout started in Midwest, then expanded to Ontario area, and finally Northeast part of United States. The electric power was not restored for four days in some area. Parts of Ontario had to experience rolling blackout for a week. It is estimated that 50 million people were affected and 61,800 MW of electric load was interrupted during the blackout. The total costs ranges between \$4 billion and \$10 billion in the United States. The final report [US 2004] of the blackout mentioned many reasons, and one of them is the failure of the alarm system of First Energy's (FE's) EMS<sup>4</sup> XA/21 software.

### **C.8.3.10.2 Software Failures**

There are many reasons indicated in 2003 Blackout final report but one of the important reasons is the malfunction of the computer systems of FE and Midwest ISO (MISO, Midwest Independent System Operator). Right before the blackout, FE's control room operators lost the alarm function that provided audible and visual indications when a significant piece of equipment changed from an acceptable to problematic condition. Shortly thereafter, the EMS system lost a number of remote control consoles. Next, it lost the primary server computer that was hosting the alarm function, and then the backup server such that all functions that were being supported on these servers were stopped. Therefore, FE's operators were unaware of that their electrical system conditions began to degrade. The alarm failed sometime shortly after 14:14 (the last time that a valid alarm came in), after voltages had begun deteriorating by well before any of FE's lines began to contact trees and trip out. Neither FE's control room operators nor FE's IT EMS support personnel were aware of the alarm failure. If an EMS's alarm are absent, but operators are aware of the situation and the remainder of the EMS's functions are intact, the operators can still possibly continue to monitor and exercise control of the grids. The FE's EMS failed either because of the stalling of the alarm processing application, "queuing" to the remote EMS terminals, or some combination of the two, which caused the server failed to complete a specific event and produce any other valid alarms (the reason of this problem is due to a software bug and explained below).

Following preprogrammed instructions, the alarm system application and all other EMS software running on the first server automatically transferred ("failed-over") onto the back-up server. However, because the alarm application moved intact onto the backup while still stalled and ineffective, the backup server failed 13 minutes later, at 14:54 EDT. Accordingly, all of the EMS applications on these two servers stopped running. The unawareness of the alarm failure and the lack of visualized display of measurement data, the FE's operators still thought their system was

---

<sup>4</sup> An EMS (Energy Management System) is a system of computer-aided tools used by operators of electric utility grids to monitor, control, and optimize the performance of the generation and/or transmission system. The monitor and control functions are know as "SCADA" (Supervisory Control And Data Acquisition). The optimization packages are often referred to as "advanced applications".

satisfactory until they received phone calls from other locations and information sources from MISO; American Electric Power (AEP); Pennsylvania, Jersey and Maryland Interconnection (PJM); and FE field operating staff. Without a functioning alarm system, the FE control area operators failed to detect the tripping of electrical facilities essential to maintain the security of their control area even after AEP identified the 14:27 EDT circuit trip and reclosure of the 345 kV line at AEP's South Canton substation. FE failed to be aware of the Harding-Chamberlin at 15:05:41 EDT or Hanna-Juniper line trips at 15:32:03 due to loss of the alarm system until MISO manually updated the state estimator at 15:41 EDT and informed FE.

Loss of the first server and the backup server caused an auto-page to be issued to alert FE's EMS IT support personnel to the problem. They did not notify the control room operators of this problem. At 15:08 EDT, IT personnel completed a "warm reboot" of the primary server. It appears the computer and all the processes were running without knowing and confirming that the alarm function was still frozen. A "cold reboot" was also considered. However, they decided not to take such an action after discussion with the operators at 15:43 EDT because the operators considered power system conditions precarious and were concerned about the length of time it might take.

After the alarm processing application of the EMS failed, the system was operated using the outdated information. Concurrently, MISO was also experiencing computer malfunctioning. The state estimator<sup>5</sup> was turned off between 12:15 EDT to 16:04 EDT in order to fix a computer glitch because it produced a highly unmatched result (the MISO state estimator was actually reactivated at 14:44 EDT but again showed a mismatch). However, it was unable to account for outages occurred when it was offline. This prevents MISO's state estimator from promptly performing the precontingency analyses and "early warning" assessments and predicting the system status due to the lack of system information. Other operating areas were not informed the system status and this caused the cascading blackout.

In February 2004, First Energy announced that the cause of the alarm failure before the 2003 blackout was a programming failure [Jesdanun 2004]. The bug was unmasked as a particularly subtle incarnation of a common programming error called a "race condition". There was a couple of processes that were in contention for a common data structure, and through a software coding error in one of the application processes, they were both able to get write access to a data structure at the same time. That corruption led to the alarm event application getting into an infinite loop and spinning. The bug was triggered by a unique combination of events and alarm conditions on the equipment it was monitoring. The utility has applied fixes developed by the system's vendor, GE, and have accelerated plans to replace GE's XA/21 with a system from French nuclear engineers Areva SA. GE distributed warning and a fix to its more than 100 other customers. This software

---

<sup>5</sup> State Estimator uses the real-time data measurements available on a subset of those power facilities in a complex mathematical model of the power system that reflects the configuration of the network (which facilities are in services and which are not) and real-time system condition data to estimate voltage at each bus and to estimate real and reactive power flow quantities on each line or through each transformer. Reliability coordinators and control area who have state estimators commonly run a state estimation on regular intervals or upon demand. Not all control areas are equipped with state estimators.

glitch was pinned by GE and energy consultants from Kema Inc., in Nov., 2003. The software bug surfaced because of a number of unusual events occurred simultaneously.

The error forcing context is a unique combination of events [Jesdanun 2002] (the details of the combination are unknown) that causes either the stalling of the alarm processing application, “queuing” to the remote EMS terminals, or some combination of the two. In this situation, the so-called “race condition” is satisfied and the alarm application went into an infinite loop and spinning.

### **C.8.3.10.3 Consequence**

50 million people were affected and 61,800 MW of electric load was interrupted during the blackout. The total costs ranges between \$4 billion and \$10 billion in the United States. In Canada, GDP was down 0.7% in August. There was a net loss of 18.9 million work hours, and manufacturing shipments in Ontario were down \$2.3 billion (Canadian dollars).

### **C.8.3.10.4 Likelihood of Error Forcing Context**

Since the EMS (and thus its alarm system) is running all the time, the likelihood of error forcing context can be represented using the occurrence frequency. More than 100 customers were using GE’s XA/21 when the blackout occurred. The failure of FE’s alarm system might not be the only one. The data of alarm system failure can possibly be collected to give the frequency of the error forcing context.

### **C.8.3.10.5 Failure Categorization**

Failure modes:

- System failure modes:
  - Halt/abnormal termination without clear message
- Element failure modes:
  - Timing/order failure (race condition) in PROCESSING element (alarm system may also be considered OUTPUT element)
  - Omission of a function or incorrect implementation of a function (it was unusual to receive no warning messages in hours) in OUTPUT element

Failure causes:

Temporal fault in System analysis and design stage (race condition was generated).  
Missing functions of alarm (desired functions are not specified in the requirements in software requirement analysis stage, i.e., no message within long time period is abnormal and the system was not able to handle this abnormally).

EFC:

Error forcing context is a unique combination of events (the details of the combination are unknown) that caused either the stalling of the alarm processing application, “queuing” to the remote EMS terminals, or some combination of the two.

Failure effects and consequences:

50 million people were affected and 61,800 MW of electric load was interrupted during the blackout. The total costs ranges between \$4 billion and \$10 billion in the United States. There was a net loss of 18.9 million work hours, and manufacturing shipments in Ontario were down \$2.3 billion (Canadian dollars).

### **C.8.3.10.6 Dependent Failure and CCF**

Yes. Both main server and the backup server run the same software. If it is stalled on the main server, it will also stalled on the backup server.

### **C.8.3.10.7 Discussion**

According to GE, patches have been installed on other EMS XA/21 after the blackout.

## **C.8.3.11 Common Cause Failure of Security Computers at San Onofre Unit 1 - 1998**

### **C.8.3.11.1 Summary**

On January 14, 1998, Southern California Edison (SCE) prepared to install a chart recorder on the primary security computer for system diagnostic testing. At about 9:25 A.M., before starting the installation, SCE had posted compensatory guards for the appropriate plant areas, as specified in Station Procedure SO123-IV-6.8, "Protected Area and Vital Area Barrier Patrols," for a complete loss of security computers. SCE switched to the backup security computer, removed the primary computer from service and installed the chart recorder. When returning the primary computer to service, a computer network server software error occurred, causing the primary computer to initialize incorrectly. At about 10:26 A.M., the backup computer also failed as a result of this error. The primary and backup computers were restarted at about 10:32 A.M. and 10:36 A.M., respectively. The cause of this event was an equipment failure. During the reboot of the primary computer, the network server function for the security computers did not start. However, the "boot" sequence continued until the main security program started on the primary computer. Without the network server function, the two computers could not completely communicate and consequently could not fully function. The main security program was not capable of recognizing that the network server function had not started and tried to regain the primary role in the security monitoring system. As a result, a conflict arose and the backup program became unstable and failed to function. Since the primary had no network server function, it could not communicate properly, leaving both primary and backup down.

This event is documented in LER 206-1998-001. The LERs that are categorized as "Safeguards/Security" are not available for general public viewing, so it was not available for this study at this time. The limited information about this LER was obtained from NUREG/CR-6734, Vol. 2 [Hecht and Hecht, 2001].

#### **C.8.3.11.2 Software Failures**

There appears to be two types of software failures in this event:

- 1) A computer network server software error. It appears that this error occurred in the primary security computer. No information is currently available on this error, so it is not further discussed here at this time.
- 2) Failure of the software of the primary and backup security computers:
  - 2.1 When returning the primary computer to service, the "main security program" started on this computer. However, this program was not capable of recognizing that the network server function was unavailable; as a result, it could not communicate properly.
  - 2.2 The "backup program" became unstable and failed to function due to the conflict that happened when the "main security program" tried to regain the primary role in the security monitoring system without the network server function.

The original cause of the failure of the "main security program" (in the primary computer) was that initialization requirements did not cover start-up from unusual conditions (in this case, network server function unavailable), causing a communication link to be dropped. The problem was compounded by lack of robustness in that requirements for neither the primary nor the back-up computer provided for verification of the presence of the communication link.

#### **C.8.3.11.3 Consequence**

The primary and backup security computers failed and were unavailable for several minutes. The impact of these failures on the security of the plant was not found in the description available. Since the licensee had posted compensatory guards for the appropriate plant areas for a complete loss of security computers according to its procedure, this impact appears to be minimal.

#### **C.8.3.11.4 Likelihood of Error Forcing Context**

The EFC for the computer network server software error was not found in the information available. The EFC for the failure of the software of the primary and backup security computers is the start-up of the "main security program" (in the primary computer) when the network server function was unavailable due to the network server software error. Since we currently do not have information available about this error, the likelihood of the EFC of the failure of the software of the primary and backup security computers is not known at this time.

### **C.8.3.11.5 Failure Categorization**

Failure modes:

- System failure mode:

Halt/abnormal termination with clear outcome, i.e., failure of primary and backup security computers.

- Element failure mode:

Possibly, the “communication” element of the “main security program” (in the primary computer) omitted the function of managing communication properly when the network server function is unavailable. Also possibly, the “communication” element of the “backup program” omitted the function of managing communication properly given the conflict that happened when the “main security program” tried to regain the primary role in the security monitoring system without the network server function.

Failure causes:

Internal causes:

The cause occurred in the stage of “Software requirements analysis.” As mentioned above, the specific cause of the failure of the “main security program” (in the primary computer) was that initialization requirements did not cover start-up from unusual conditions (in this case, network server function unavailable), causing a communication link to be dropped. The problem was compounded by lack of robustness in that requirements for neither the primary nor the back-up computer provided for verification of the presence of the communication link.

EFC:

- The EFC for the computer network server software error was not found in the information available. The EFC for the failure of the software of the primary and backup security computers is the start-up of the “main security program” (in the primary computer) when the network server function was unavailable due to the network server software error.

Failure effects and consequences:

- The primary and backup security computers failed and were unavailable for several minutes. The impact of these failures on the security of the plant was not found in the description available. Since the licensee had posted compensatory guards for the appropriate plant areas for a complete loss of security computers according to its procedure, this impact appears to be minimal.

### **C.8.3.11.6 Dependent Failure and CCF**

A dependent failure occurred because the primary and backup security computers failed.



### **C.8.3.11.7 Discussion**

This event illustrates the potential of a software failure to propagate into the failure of redundant computers, i.e., main and backup.