International Task Force on Safety Critical Software (TF SCS)

Consensus positions including NRC

Mark Bowell (ONR, UK)
Pierre-Jacques Courtois (Bel V, Belgium)

Overview

- NRC publication
- Safety demonstration
- Safety plan
- Pre-existing software
- Software tools
- Formal methods
- Software design diversity
- Smart sensors and actuators
- Computer based system requirements

NRC publication – NUREG/IA

- NRC has participated in task force since 2009
- NUREG/IA published 2014
- "NRC considers the common positions a valuable technical reference for future improvements in its own regulatory guidance"
- Includes full common position document
- NRC reviewed common positions
  - changes made
- Excludes chapters on
  - system classes and function categories
  - graded requirements
  - security
  - software reliability
- NRC notes cover
  - exclusions
  - differences in regulatory framework
  - differences in vocabulary

Safety demonstration

- Justifies the properties of a particular system operating in a specific environment
- Rule based or goal based (risk has been reduced sufficiently)
- Design, plus operation, maintenance, modification and even decommissioning
- A living document evolving with the project, eg
  - as assumptions are tested,
  - modifications (including to other related systems),
  - incidents occur,
  - changes in wider safety case
- Claims, arguments and evidence
- Clarifies the safety requirements and the system properties needed to support them

Safety plan

- Overall aim:
  To show how will safety be demonstrated
- What software is to be included
- Coherent argument based on evidence
- Types of evidence
  - process
  - product
  - staff competence
- What activities need to be carried out
  (to produce the demonstration)
- Organisational arrangements for these activities
- Reference standards, regulations etc

Pre-existing software

- Examples:
  Operating systems
  Device drivers
  Software libraries
  Application logic subroutines
- Problems:
  Imprecise specification
  Little or no development process evidence
  Operational experience insufficient to rigorously demonstrate safety

Pre-existing software

- Specify requirements for safety system context
- Show consistency between these requirements and overall safety requirements
- Rigorous version control
- Validate interfaces
- Test behaviour in context of safety requirements
  - un-required functionality cannot compromise safety functions
- Compliance of the design with applicable standards requirements
- Evidence of good quality commercial practice in software development
- Take care with arguments based on operational experience
  - effective error reporting
  - operational profile
  - identical software versions or demonstrated equivalence

Software tools

- Examples:
  Requirements engineering
  Design modelling
  Transformation (Complier, Code generation)
  Automated dynamic testing
  Static analysis
  Theorem proving
  Documentation
  Change control
- Advantages:
  Reduce some types of error
  Standards adherence
  Consistent and comprehensive records
- Problems:
  Common cause errors
  Application limits

Software tools

- Determine qualification requirements
  - impact on target software
  - detection of faults by other tools
  - qualification of combined set to a level that preserves the required properties
- For safety systems
  - manufacturer's QA documentation available
  - verification of output and validation on target system unless special justification gives evidence that
    - tool accepts only input data that is syntactically valid
    - tool preserves the semantics of valid input through to output
- Impact analysis of version changes
- Dissemination of feedback from other users

Formal methods

- Safety demonstration "credit" must be explicitly linked to
  - its contribution (what is being argued)
  - evidence that argument is justified
- Document system boundaries and non-functional properties
- Justify the combination of methods used
- Demonstrate that techniques used have relevant pedigree
- Document procedures and constraints for use
  - explicitly document limitations
- Validate formal description of system requirements against prior plant safety analysis

Software design diversity

- Objective:
  To avoid common cause failure
- Justify using or not using diversity
  - show how the techniques avoid common cause failure
- Diversity cannot compensate for a lack of quality
- Functional diversity as a primary means
- Specifically analyse systems for common cause failure

Smart sensors and actuators

- A type of pre-existing software
  - closer relationship to supplier
  - difficulties in access to design documentation
  - usually not nuclear-specific
  - hidden changes
- Clearly identify:
  make
  model
  application
  software versions
- Demonstrate fitness for purpose
- Safety class – essential part of system
  - specify the properties required
- Identify failure modes
- Analyse common cause failure

Smart sensors and actuators

- Compare production to an applicable safety standard
  - address deficiencies by compensating activities or arguments
    - manufacturer rework
    - static and dynamic testing
    - restrict functionality
- Confidence building independent of the supplier
  - commissioning tests
  - static analysis
  - analysis of operating experience
  - statistical testing
- Document safe envelopes and limits for acceptable use
  - functionality
  - maintenance
  - environment

Computer based system requirements

- Requirements are foundational, based on
  - plant safety analyses
  - applicable regulations
- Co-operative endeavour
  - trade-off between formality and accessibility
- System requirements traceable to
  - the results of prior plant safety analyses and
  - other relevant analyses at the plant level
  - in the context of the
    - safety plan
    - safety demonstration
    - system descriptions
- Documented results of validation based on traceability

Computer based system requirements – will include

- System boundaries, interfaces, constraints
- Non-functional requirements (ISO 25000 "quality requirements")
- Required responses to hazardous conditions
- Defences against potential incorrect system outputs
- Relationships between input and output variables, including valid ranges
- Limits of configuration, eg calibration constants, setpoints
- All modes of operation
  - at power, refuelling, post accident monitoring
- All human interactions
  - operation, maintenance, tests
- Notification of correct completion to the operator