Acceptance Criteria for Systems Supported by an Assurance Case

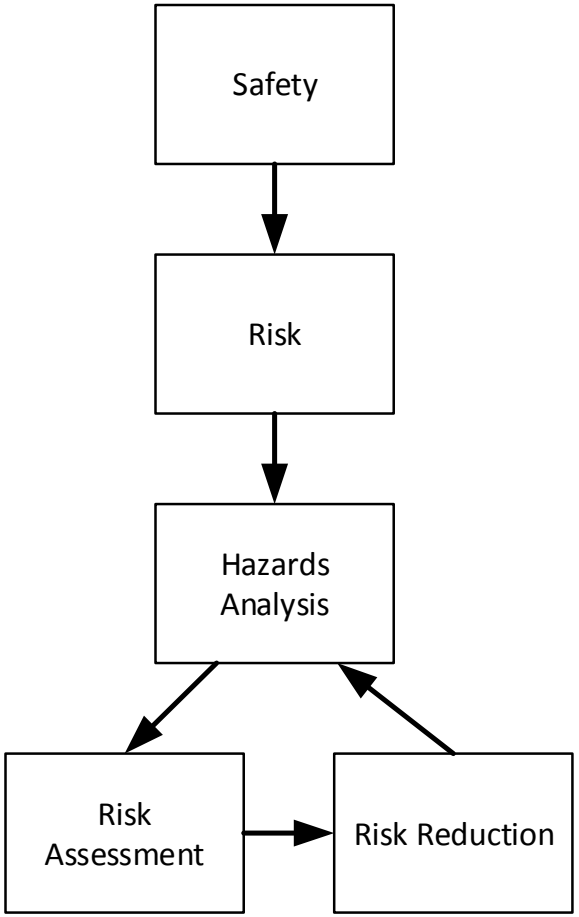
Paul Joannou
McMaster Centre for Software Certification

July 18, 2015

Introduction

- Experience from nuclear industry in Canada
- Regulatory consensus on “How good is good enough?” aka Acceptance Criteria
- Relationship to assurance cases

Risk Based Definition of Safety



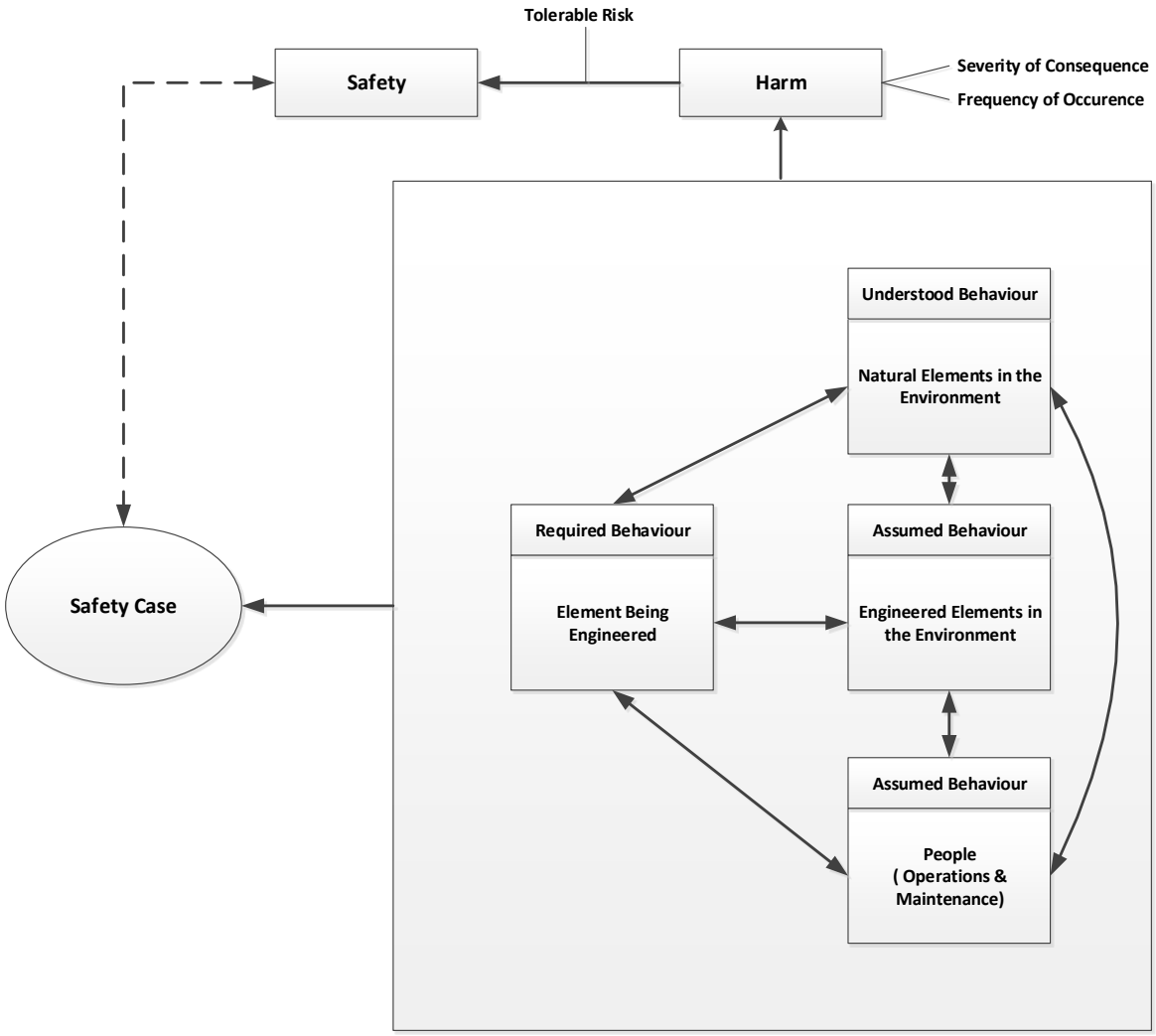
- freedom from unacceptable risk

- severity of consequences
- likelihood of occurrence

- identify hazards
- analyze risks associated with each hazard
- reduce risks to tolerable levels

- FTA
- FMEA
- Event Tree Analysis
- Common Cause Failure Analysis
- Cause Consequence Analysis
- STPA

Scope of a Safety Case



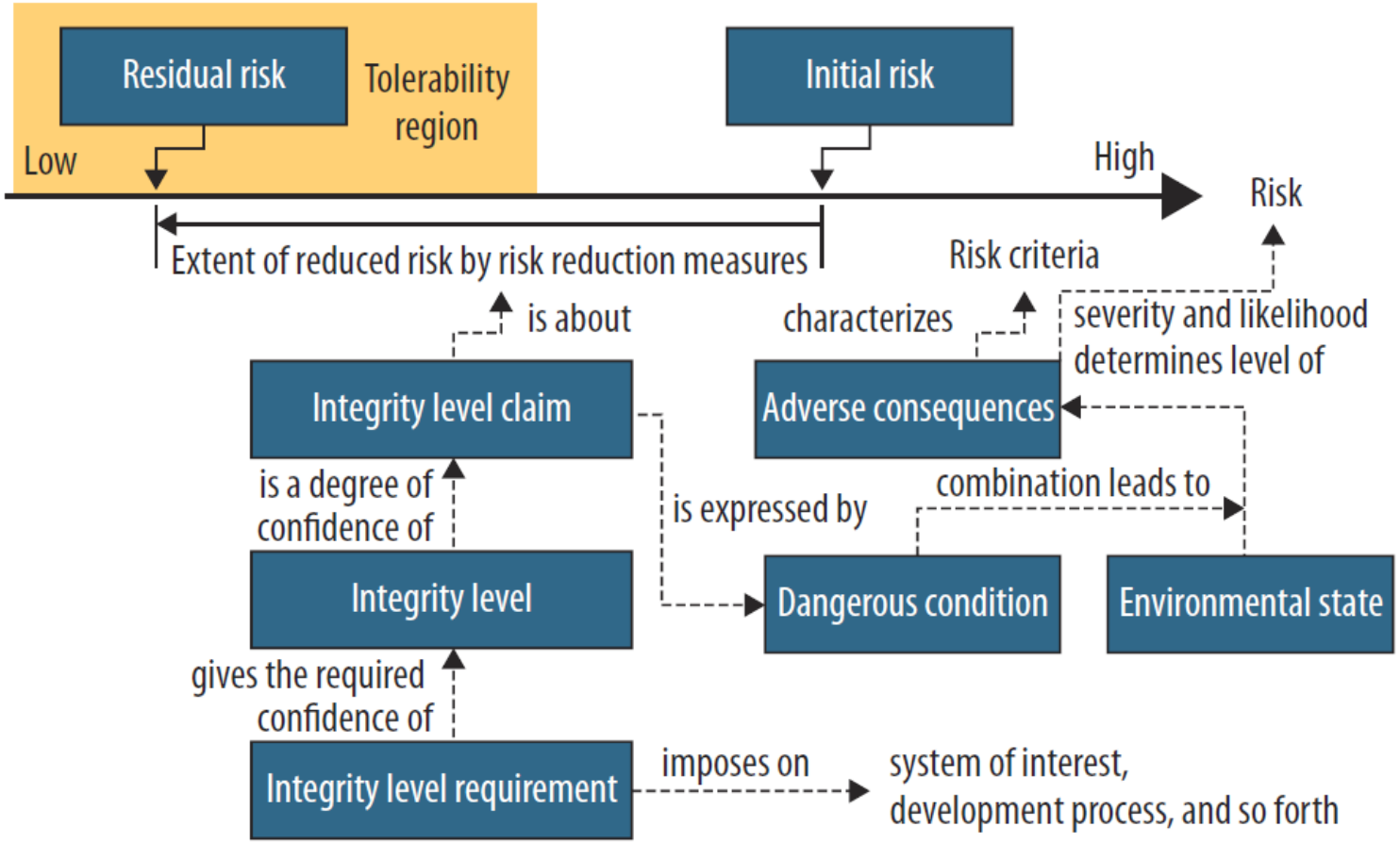
Confidence that the claims of the system being engineered are correct and complete,

Confidence that the system has been implemented consistent with the claims,

Confidence that the system will be operated and maintained consistent with design assumptions, and

Confidence that 1, 2 and 3 above are sustained during the lifetime of the system.

Required Degree of Confidence (Integrity Level Terms)



From draft revision of ISO/IEC 15026 - Systems and software assurance — Part 3: System integrity levels

CANDU Standard for Software Engineering of Safety Critical Software

- Developed in response to regulatory issues with digital safety systems
- Provided basis for regulatory consensus on acceptance criteria
- Defines methodology independent requirements
- Since initial issue in 1990 has been found to provide a practical and effective approach for the development of safety critical software

ONTARIO POWER GENERATION INC.

AECL Atomic Energy of Canada Limited EACL Énergie atomique du Canada limitée

CANDU® COMPUTER SYSTEMS ENGINEERING CENTRE OF EXCELLENCE

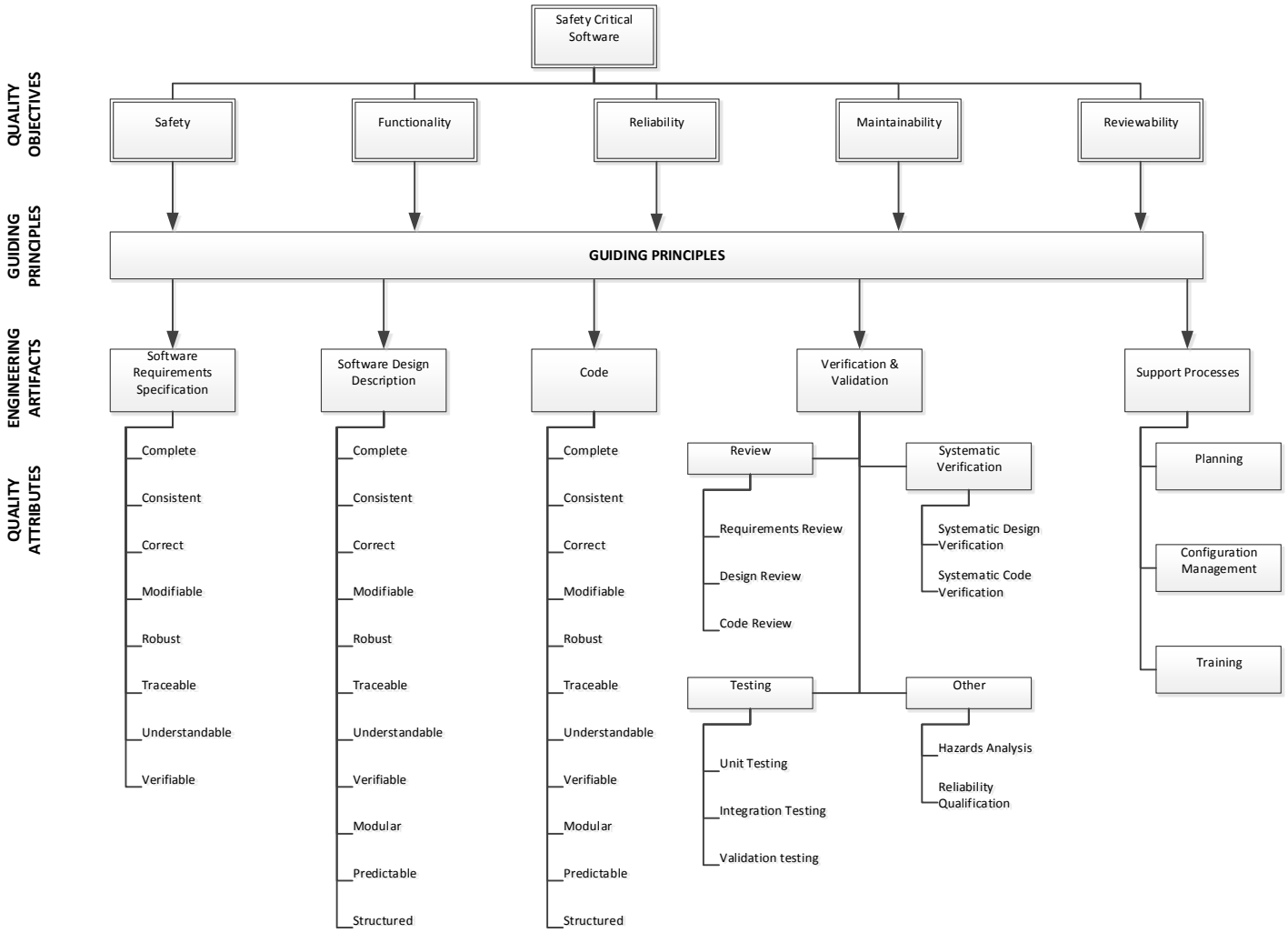
STANDARD

CE-1001-STD
Revision 2

Standard for
Software Engineering of
Safety Critical Software

December 1999

Standard for Software Engineering of Safety Critical Software



Claims

Argument

Evidence

Quality Objectives (claims)

Safety: A quality objective that requires that the system function in accordance with its *requirements*, in a consistent and predictable manner, under all conditions. When the system can no longer perform its required role, this quality objective requires that it act to maintain the equipment and processes it controls in a safe state in all situations.

Functionality: A quality objective that requires that the system implement all required behaviour and meet the performance requirements.

Reliability: A quality objective that requires that the system perform its required behaviour such that the probability of it successfully performing that behaviour is consistent with the reliability requirements identified.

Maintainability: A quality objective that requires that the system be structured so that those items most likely to require modification can be changed reliably and efficiently. This quality objective also requires the rationale for design decisions be evident to a third party.

Reviewability: A quality objective that requires that the system be developed and documented so that it can be systematically inspected by a third party for conformance to requirements.

Principles (argument)

- The **required behaviour** of the *software* shall be documented. Mathematical functions in a notation that has well defined *syntax* and *semantics* shall be used.

- The **structure of the software** shall be based on information hiding and the use of recognized software engineering practices

- The outputs from each development process shall be reviewed to **verify** that they comply with the requirements specified in the inputs to that process. Mathematical verification techniques or rigorous arguments of correctness shall be used to verify that the code meets the required behaviour specified in the SDD, which meets the required behaviour specified in the SRS.

- Verification of the software shall be carried out throughout its entire life. Any **changes** shall be verified to the same or better degree of rigour as the original development. The extent of verification necessary is determined by an analysis of the scope of the change.

- **Independence** of design and verification or validation personnel to the extent required shall be maintained to help ensure an unbiased verification or validation process.

Principles (argument)

- Engineering of the software shall follow **a planned and systematic process** over the entire life of the software. All activities in the software life cycle shall be carried out using approved procedures and guidelines that conform to the intent of this standard.
- **Configuration management** shall be maintained throughout the entire life of the software to ensure up-to-date and consistent software and documentation.
- Training shall be undertaken to ensure that **personnel** have the knowledge, skills and attitudes required to perform their jobs.
- Behaviour over the **full range** of possible values of monitored variables shall be specified and implemented.
- All **errors** reported from interactions with hardware or predeveloped software shall be checked and handled.
- The software design shall be as conservative and **simple** as possible, while meeting all requirements and retaining spare capacity for anticipated changes.

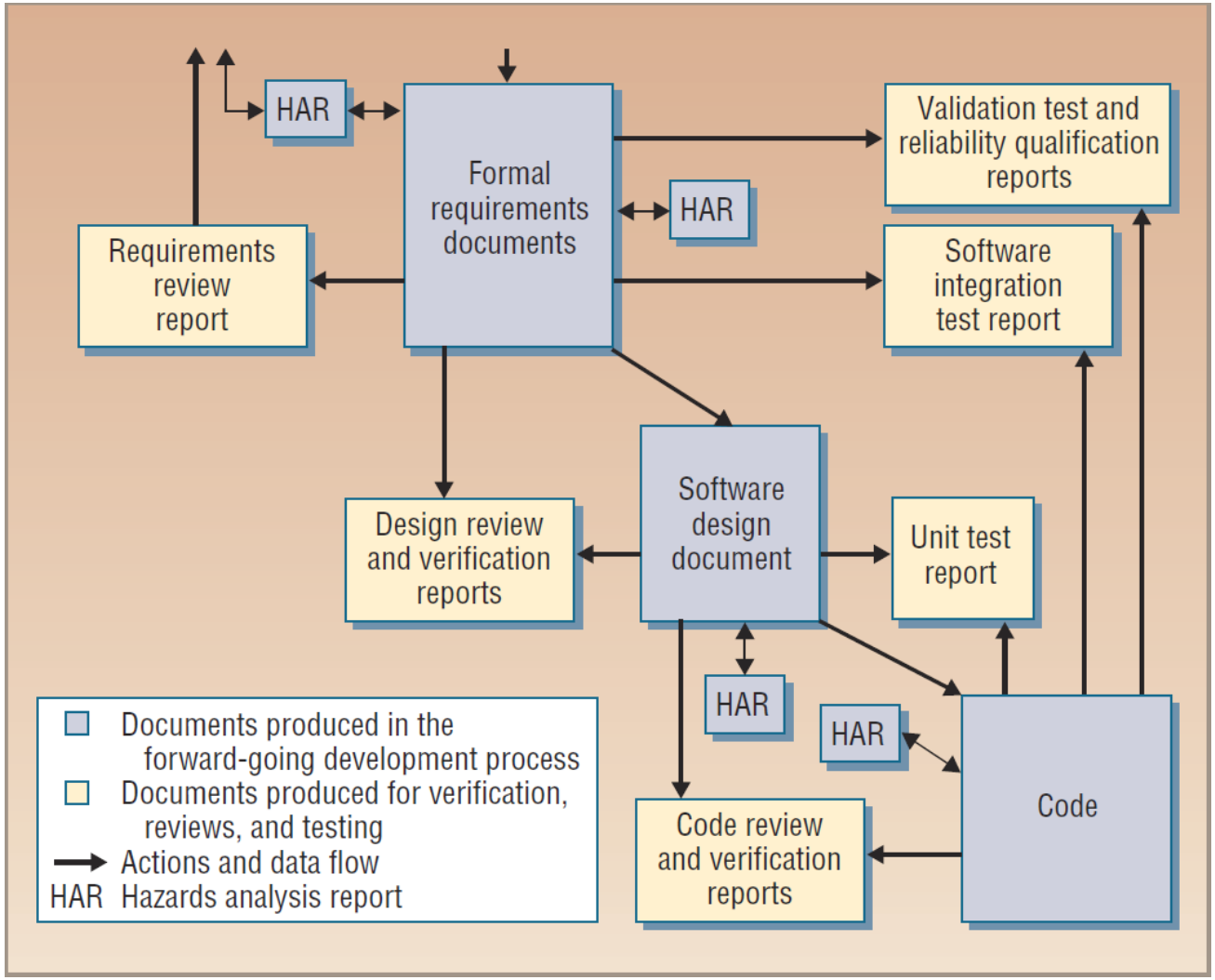
Principles (argument)

- Analyses shall be performed to identify and evaluate safety **hazards** associated with the computer system with the aim of either eliminating them or assisting in the reduction of any associated risks.

- Reliability of the safety critical software shall be demonstrated using a **statistically valid, trajectory-based methodology** in which a reliability hypothesis is proven to the degree of confidence required.

- **Adequate testing** shall be performed to meet predefined test coverage criteria.

Evidence *(plus planning, CM & training)*



Set of Acceptance Criteria for Each Artefact

Requirements on the SRS		completeness	consistency	correctness	modifiability	modularity	predictability	robustness	structuredness	traceability	understandability	verifiability
3.1.2.a	- Specify all requirements from the DID	√										
3.1.2.b	- Specify additional requirements	√		√								
3.1.2.c	- Describe context										√	
3.1.2.d	- Identify external properties	√									√	
3.1.2.e	- Describe I/O characteristics	√										
3.1.2.f	- Describe I/O relationships	√										
3.1.2.g	- Define behaviour	√		√							√	√
3.1.2.h	- Specify exception response	√						√				
3.1.2.i	- Specify fault tolerance							√				
3.1.2.j	- Specify timing tolerances	√										
3.1.2.k	- Identify anticipated changes	√			√							
3.1.2.l	- Identify design constraints	√										
3.1.2.m	- Requirements and constraints only	√			√							
3.1.2.n	- No conflicts		√	√							√	√
3.1.2.o	- Unique requirements		√		√							
3.1.2.p	- Identify requirements uniquely									√		
3.1.2.q	- Reference design notes									√		
3.1.2.r	- Demonstrate mapping									√		
3.1.2.s	- Conform to standards	√	√	√								
3.1.2.t	- Use consistent terminology		√								√	√
3.1.2.u	- Enable role determination			√	√						√	√

Purpose of an Assurance Case

- **Assurance**: grounds for justified confidence that a claim has been or will be achieved

- **Assurance case**: a reasoned, auditable artefact created that support the contention that its top-level claim (or set of claims), are satisfied, including systematic argumentation and its underlying evidence and explicit assumptions that support the claim(s)

- **Approval authority**: the person (or persons) and/or organization (or organizations) responsible for approving activities, artefacts, and other aspects of the system during its life cycle

Acceptance Criteria

- The acceptance criteria for providing adequate confidence is based on:
  - Qualifications of the individuals involved,
  - Use of due process in a planned and systematic manner,
  - Clear documentation that is third party reviewable, and
  - Objective evidence that the system meets its quality objectives.

- Acceptance criteria allow various methods to be used to achieve the acceptance criteria and hence do not unnecessarily constrain the methods

- Approach has resulted in:
  - practical and effective means to develop safety critical software,
  - a stable backdrop against which to continuously improve methods and tools and
  - has provided regulatory predictability

Summary

- Appropriate scope for the assurance case
  - Correctness & completeness of claims
  - Implementation consistent with claims
  - Operation & maintenance consistent with design assumptions
  - Maintenance of assurance case over life of system
- Integrity level claims vs integrity level requirements
- Level of confidence
- Acceptance criteria for approval of an assurance case
- Research challenge: scientific basis for the acceptance criteria

References

1) Standard for Software Engineering of Safety Critical Software, CE-1001-STD, Revision 2, CANDU Computer Systems Engineering Centre of Excellence, December 1999
http://shop.csa.ca/content/ebiz/shopcsa/resources/documents/nuclear/CE-1001-STD%20Revision-2.pdf

2) Book Section 1995, Safe Comp 95, “Ontario Hydro’s Experience with New Methods for Engineering Safety Critical Software”, Viola, M.

3) Maibaum T S E & Wassyng A, “A Product-Focused Approach to Software Certification”, column for IEEE Computer – Software Technologies, IEEE CS Press, Feb 2008

4) Joannou P & Wassyng A, “Understanding Integrity Level Concepts”, column for IEEE Computer – Standards, IEEE CS Press, Nov 2014