DRAFT RESEARCH INFORMATION LETTER 1101:
Technical basis to review hazard analysis of digital safety systems

1 Executive Summary

This research information letter (RIL) provides the US Nuclear Regulatory Commission (NRC)'s licensing staff the technical basis to support the exercise of judgment in their review of hazard analysis (HA) performed on a digital safety system by an applicant seeking design certification or a license amendment.

The RIL is prepared in response to a user need request from the Office of New Reactors (NRO), dated December 8, 2011, asking the Office of Nuclear Regulatory Research (RES) for assimilation of the technical basis to support regulatory review of an applicant's HA relevant to digital instrumentation and control (DI&C) safety systems in nuclear power plants (NPPs). NRC does not have explicit guidance on review of HA; therefore, NRO intends to use this RIL to develop and support review guidance for piloting in a project proposing new digital technology for a small modular reactor. From this learning cycle, NRC expects to identify needs for future improvements in its review guidance, regulatory guidance, and the underlying technical basis. Thus, this RIL is intended for these early adopters.

The RIL has been focused on issues encountered in NRO's recent licensing reviews – particularly hazards which are rooted in systemic causes such as inadequacies in engineering; these causes are called contributory hazards in the RIL. The technical basis is focused on evaluation of an applicant's HA rather than performing HA.

Digital safety systems are becoming more difficult to analyze and evaluate for safety, due to rapid changes in the nature of systems and the underlying technologies, increasing inter-connectivity and interactions across systems, and resulting shortening of relevant accumulated experience. Unwanted interactions and side effects are becoming significant contributors to hazards in many critical application domains of digital technology. In contrast to a hardware failure event, which, typically, occurs in a particular component, engineering deficiencies in complex digital systems tend to be pervasive in nature. Traditional techniques of hazard analysis (e.g, failure modes and effects analysis; fault tree analysis; event tree analysis) are rendered ineffective when the causes are systemic and pervasive. Typically, uncertainty from such hazards is best addressed by rooting out the causes. However, often, the causes are hard to find. RIL-1101 identifies some common hazard contributing scenarios and, for each, examples of conditions that reduce the respective hazard spaces. These cause-effect relationships form the core of the technical basis in RIL-1101, which the staff assimilated from existing knowledge through a combination of literature search and expert-consultation. These causal relationships also serve as a safety goal focused organizing framework for an applicant's analysis, whether it is for a new system in a new reactor or modification of a module in an existing system.

The broader hazard analysis approach covered in RIL-1101 can be applied first on an early-stage functional concept and iterated as the development progresses on the successive work products. The resulting design criteria and design bases include constraints to avoid conditions contributing to hazards. Identification of such conditions early in the development lifecycle to drive subsequent engineering helps avoid problems downstream. It not only reduces uncertainty about safety, but also improves lifecycle economics.

Contentsc

Figures

Page #

Tables

2 Introduction

This research information letter (RIL) provides the US Nuclear Regulatory Commission (NRC)'s licensing staff the technical basis to support the exercise of judgment in their review of hazard analysis (HA) performed on a digital safety system by an applicant seeking design certification or a license amendment. Section 2.5 provides a brief background on HA, supported with elaboration in Appendix C. Section 2.6 states the purpose and intended audience.

2.1 Regulatory basis

Hazard analysis of a digital safety system could address clauses 4.8 and 5.6 in [3] and the analysis aspect of 10 CFR 50.34(a)(3) [35] and 10 CFR 52.47(a)(2) [36]. In support of requirements in 10 CFR 50.34(a)(3)(i) and 10 CFR 52.47(a)(3)(i), hazard analysis could support developing principal design criteria. In support of requirements in 10 CFR 50.34(a)(3)(ii) and 10 CFR 52.47(a)(3)(ii), hazard analysis could lead from principal design criteria to design bases. Section 4 discusses the regulatory significance of this work further.

2.2 Work authorization

The RIL is prepared in response to a user need request from the Office of New Reactors (NRO), dated December 8, 2011, asking the Office of Nuclear Regulatory Research (RES) for assimilation of the technical basis to support regulatory review of an applicant's HA relevant to digital instrumentation and control (I&C) safety systems in nuclear power plants (NPPs). The user need arose, because NRC does not have explicit guidance to review HA for a digital safety system of the kind seen in recent licensing reviews.

2.3 Relationship with licensing experience

The RIL has been focused on issues encountered in NRO's recent licensing reviews – particularly hazards, which are rooted in systemic causes such as inadequacies in engineering; these causes are called contributory hazards in the RIL. The technical basis is focused on evaluation of an applicant's HA rather than performing HA. Thus, the RIL is not intended to be a self-contained, comprehensive, and complete stand-alone technical reference for reviewing HA of digital safety systems in NPPs. Section 2.7 elaborates the scope. Section 2.8 explains the organization of the RIL.

Digital safety systems are becoming more difficult to analyze, due to many factors, such as the following:

- Rapid changes in the nature of systems and the underlying technologies. (H-OTproc-7)
- Increasing inter-connectivity. (Section 3.4 H-ProcState-5)
- Resulting shortening of accumulated experience relevant to a new system. (H-OTproc-7)

Examples of associated contributory hazards include the following:

- Inadequately constrained interactions of the digital safety system being analyzed with other systems and elements in its environment.
- Incorrect decomposition and allocation of NPP-level safety functions into NPP-wide I&C architecture and then to the digital safety system being analyzed.

- Inadequate identification of the quality properties[1] (e.g.: safety assurability; verifiability; analyzability) associated with safety functions.
- Incorrect flow-down into constraints on the architecture of the system and then the architecture of the software or other forms of logic.
- Inadequate flow-down to identify requirements and constraints on technical processes, supporting processes, organizational and processes.
- Declining supply and replenishment of requisite competence. (Section 3.1 H0-2).
- Longer less track-able supply chains. (Section 3.1.2 item 2 under note for H0-9)
- Inadequate quality of cross-organizational cross-disciplinary communications, etc. (Section 3.2 H-culture-9)

2.4 Significance of the technical basis in licensing reviews

For each "contributory hazard scenario" (which illustrates some hazard space[2]) the RIL provides examples of conditions that reduce the hazard space. These cause-effect relationships form the core of the technical basis in RIL-1101, assimilated from existing knowledge, acquired through a combination of literature search and expert-consultation. These causal relationships also serve as a safety goal focused organizing framework for an applicant's analysis.

To suit project-specific needs, NRC's licensing offices can select "contributory hazard scenarios" and corresponding conditions to reduce the respective hazard spaces, and transform these conditions into review criteria; NRO's mPower design specific review standard (DSRS) Appendix A [1] is an example.

2.5 Background

A hazard, in general, is defined as "potential for harm." In RIL-1101, the scope of "harm" is limited to the degradation of the performance of an NPP safety function allocated to the system to be analyzed.

Hazard analysis (HA), a systems engineering activity[3], is the process of examining a system throughout its lifecycle to identify inherent hazards and contributory[4] hazards, and requirements and constraints to eliminate, prevent, or otherwise control them.

HA is a subset of safety analysis; its evaluation is a subset of safety evaluation – the relationship is explained in Section 2.7.8.

Current practice exhibits a wide variation in usage of the terms, hazard and hazard analysis. For example, some experts distinguish between a hazard, its source, and its cause. To avoid confusion, RIL-1101 bounds the scope of HA as follows:

[1] In common practice, these are treated as "non-functional" requirements.

[2] Hazard space: All the possible combinations of specific conditions, relevant to a scenario that could lead to the degradation of a safety function.

[3] This implies use of systematic and replicable methods in performing HA.

[4] It includes causal factors

1. NPP-level safety analysis (including NPP-level HA[5]) identifies functions required for NPP-level safety (known as safety functions) and correctly identifies the functions to be allocated to the I&C level.
2. All hazards leading to the degradation of a safety function allocated to the I&C level are identified.
3. Causes, including contributory causes, (collectively known as contributory hazards) are identified.
4. Commensurate requirements and constraints[6] are identified.

2.6 Purpose and intended audience

The purpose of this research information letter (RIL) is to provide the technical basis to support NRC I&C staff in the exercise of judgment during licensing reviews they[7] perform on an applicant's hazard analysis (HA) of a digital safety system in a nuclear power plant (NPP).

Since NRC does not have any relevant explicit guidance on review of HA, this RIL is intended for NRO's early adopters, to support their development of review guidance to be piloted in a new project applying new technology in a digital safety system for a small modular reactor. This application will serve as a learning cycle, from which NRC expects to identify needs for future improvements in its review guidance, regulatory guidance, and the underlying technical basis (i.e., successors to RIL-1101).

The RIL is not intended as an interim or surrogate regulatory guide to licensees or applicants. However, as a technical basis for the limited scope described in the next subsection, it may also be useful to stakeholders outside the NRC.

2.7 Scope

The RIL is a response to NRO's user need request for supporting a specific project. However, the content is sufficiently generic to evolve a successor for broader application, after learning from NRO's first experience [Section 2.7.1]. Content has been selected to support evaluation rather than performance of HA [Section 2.7.2] for NPP safety automation [Sections 2.7.3-2.7.5]. Content is focused on hazards contributed through systemic causes, especially inadequacies in engineering [Section 2.7.6]. Content is focused on supporting a deterministic review process [2.7.7].

2.7.1 Immediate scope limited to learning cycles

Although the content provided in RIL-1101 is intended to be more broadly applicable, the adequacy for broader application has to be validated through experience. Known limitations are identified below.

[5] The technical basis for evaluating NPP-level HA is outside the scope of RIL-1101. The interactions between a digital safety system and its environment (the plant) are within scope.

[6] Specifically, in its scoping of HA, RIL-1101 leaves the creation of constraint-satisfying solutions to the primary development activities. See Appendix C.2.

[7] It includes their agent or a third party

2.7.1.1 Assumptions about areas not well understood

Within the scope described above, RIL-1101 focuses on areas that are not well understood or recognized (e.g., those rooted in systemic causes and contributed through engineering deficiencies in system development). To quote from [2]:

> "Common underlying factors[8] involve organizational culture, safety culture, fatigue, other fitness for duty issues, training, experience, habit, habituation, dysfunctional schedule pressure, adverse ambient conditions, work-related distractions, and the like. Nevertheless, addressing ineffective hazard recognition instances, addressing the factors that resulted in them, and addressing their extents would be a highly cost-effective initiative."

Judgment used in the selection of coverage of the subject matter is based on assumptions about what is not well understood. Such assumptions should be re-evaluated through learning cycles, before broader application of RIL-1101.

It is assumed that hazards internal to the DI&C system, contributed by hardware elements are well understood. Therefore, review of hardware-related HA is addressed in Section 3.7 only briefly[9].

2.7.1.2 Extrapolation from recent licensing experience

Subject matter (e.g., contributory hazard scenario) was selected in consideration of issues experienced by the licensing offices in the last several review projects, with the assumption that those issues were indicative of a trend. It is possible that new issues[10] surface in upcoming reviews that were not explicitly addressed in RIL-1101. Its adequacy should be tested through several learning cycles, before broader application.

2.7.1.3 Support for application-specific customization of the SRP Chapter 7

Selection and extent of treatment of subject matter is further narrowed to support customization of the SRP Chapter 7 specific to the needs foreseen for the mPower project.

2.7.2 Focus on evaluation rather than performance of hazard analysis

RIL-1101 is focused on providing the technical basis for exercising judgment during licensing review activities. RIL-1101 is not intended as an interim or surrogate regulatory guide to licensees or applicants. RIL-1101 is not intended to provide guidance on how to perform HA.

Prevalent public standards and guides on HA elaborate on techniques to perform HA, but there is little information available on criteria for evaluating the results of HA, even though the systematization of hazard analysis is over four decades old.

2.7.3 Focus on licensing reviews of safety automation

Although results from HA, in general, include requirements for aspects outside the initially commissioned DI&C safety system (e.g., training, maintenance, and operational and maintenance environments), RIL-1101 does not provide the technical basis to evaluate requirements concerning operation and maintenance and the people engaged therein.

---

[8] RIL-1101 scope does not include all the quoted factors.

[9] Appendix C leads to more information through links to supporting references.

[10] Example: Ha

In keeping with the scope of the SRP Chapter 7, the scope of RIL-1101 is limited to the safety automation. The human, the human-automation interface, and the associated control room are treated as part of the environment (Section 3.4.1) of the system in scope.

2.7.4 Focus on safety related systems for NPPs

Prevalent public standards [3] and guides [5], [6], and [7] on HA are oriented to the general case of a system implementing a variety of functions with varying degrees of criticality. In contrast, RIL-1101 focuses on safety related systems for NPPs, where the consequence of a mishap, unwanted release of radioactivity into the environment (known in HA vocabulary as the loss), is of the highest degree of severity. The scope includes a system realizing a safety function, as well as any system or element on which the correct timely performance of a safety function is dependent (see Appendix K).

Review of analysis for hazards external to the DI&C system, in general, is covered in other parts of NRC's standard review plan [8]. RIL-1101 considers external hazards primarily from the perspective of issues with interfaces and interactions that can affect a safety function allocated to the system being analyzed.

RIL-1101 does not elaborate on reviewing the analysis of hazards from the physical environment (Section 3.4.1; Appendices E.4 and E.5)), because these are not new considerations.

2.7.5 Types of systems intended in scope

RIL-1101 describes the evaluation of an applicant's HA associated with digital safety systems for new and advanced reactors. The scope of this RIL is limited to a system realizing a safety function or on which the correct timely performance of a safety function is dependent (see Appendix K). Other elements interfacing with, interacting with or affecting the DI&C safety system are treated as parts of its environment; to that extent, such environment is also within the scope (see Section 3.4.1).

The scope treats any change to a previously analyzed DI&C safety system as a new hazard analysis review cycle.

2.7.6 Focus on contributory hazards rooted in systemic causes

The RIL is focused on hazards rooted in systemic causes such as inadequacies in engineering (elaborated in Sections 3.1-3.6 and 3.8-3.9).

Systemic causes are a special kind of common causes of failure[11] (CCF), such that, often, their propagation is pervasive; that is, there could be many propagation paths, and these are not easy to discover and analyze. (In contrast, the propagation path from a CCF due to the breakdown of a component in a hardware system is relatively easier to identify and analyze). In a system with complex logic[12], recognizing and understanding the cause-effect relationships or influence paths well enough requires explicit identification of a variety of dependencies (see Appendix K). Some dependencies can be recognized in the analysis of the system itself (e.g., Sections 3.4.2, 3.6.1, 3.6.3). Some can be recognized through analyzing interactions of the system with its environment (e.g., Section 3.4.1). Many other dependencies occur through organizational processes (e.g., Section 3.2), technical processes (e.g., Sections 3.3, 3.4.3, 3.5,

[11] Meaning in this context: Loss of the top-level safety goal.

[12] For example, in the form of software.

3.6.2, 3.6.4), and supporting or auxiliary processes. RIL-1101 does not enumerate all contributory factors and relationships exhaustively, but uses examples: Scenarios to illustrate certain hazard spaces and examples of related conditions that reduce the respective hazard spaces. These relationships are causal dependencies, known in the respective underlying scientific disciplines and have been validated through expert reviews of RIL-1101.

2.7.7 Scope excludes risk quantification

Given the focus on hazards rooted in systemic causes, the scope excludes quantification[13] of severity of consequence and probability of occurrence[14] for the following reasons:

1. The consequence of the failure of a safety function is treated at the highest level of severity.
2. Logic[15] leading to a safety function must execute correctly or the consequence is of the same level of severity as the DI&C safety system - no mitigation is possible.
3. A safety system in an NPP is an independent layer of defense; no credit for meeting the allocated safety requirements is assumed from another layer of defense[16].
4. Contributory hazards originating in the system development lifecycle or rooted in systemic causes[17] are pervasive (permeating) in their effects. The governing variables are not sufficiently controlled in the current state of practice even to identify the contributors, their contribution paths, and the effects of their interactions. The relationships of the systemic causes to the degradation of a safety function are not linear.
5. Since design certification for DI&C platforms, tools, processes, etc. allows multiple future applications, the same elements could be replicated for different NPP functions, multiplying vulnerability to the same contributory hazard. This multiplication effect is not bounded.

2.7.8 Relation between hazard analysis and safety analysis

Hazard analysis is an intrinsic part of safety analysis (see Appendix C.2).

Figure **1** shows the relationship of HA[18], as treated in RIL-1101, to other activities contributing to the applicant's safety analysis report (SAR), as explained below:

1. The result of HA activities (depicted in the upper left sector of Figure **1**) is a set of safety requirements and constraints (included in the design bases), which are verifiable independently by a third party not involved in the development of the safety system. Also included are derived requirements and constraints on the design and implementation of the safety system. This set of requirements and constraints is intended to be a part of the licensing basis.
2. Activities in the scope of inspection, tests, analyses, and acceptance criteria (ITAAC) (depicted in the upper right sector of Figure 1) verify that these requirements and constraints

[13] Scope also excludes qualitative classification or gradation.

[14] Exception: Section 3.7 pertaining to hardware components.

[15] Example: Software

[16] An independent layer of defense protects against the unknowns and uncertainties in the other layers of defense.

[17] The focus of RIL-1101

[18] Figure 1 is a simplified depiction, See note.

have been satisfied. These verification activities are not a part of reviewing hazard analysis, as delineated in RIL-1101.

3. Figure 9 shows the relationships of HA activities with mainstream system development activities and verification activities.
4. Whereas each verification activity yields corresponding evidence (e.g., that a certain item, such as hardware or firmware or software has met the requirements and constraints allocated to it), overall verification includes the integration of all the various evidence items (depicted in the lower sector of Figure 1) in a way that demonstrates that the overall safety requirements and constraints of the system have been satisfied. These activities are also not a part of hazard analysis, as delineated in RIL-1101.
5. Safety analysis (SAR), depicted by the circle in the center of Figure 1, includes the validated results of HA (i.e., validation that safety requirements and constraints have been identified correctly, completely, consistently, and unambiguously), as well as the results of verification (i.e., the former have been satisfied).

Note: Figure **1** is simplified for illustrating the relationship with the overall safety analysis, omitting the following:

1. HA is iterated at each phase in the development lifecycle of a system (see Figure 9) and the development lifecycle of each of its elements.
2. Iteration at any phase may reveal that the phase has introduced a new hazard.
3. The corrective action may simply be a revision within that phase or it may require a change in a preceding phase, invalidating the result of the preceding phase.
4. The latter case may require multiple iterations and tradeoffs, making the analysis correspondingly more difficult.
5. V&V activities during the mainstream system development are also iterative (discovery of anomaly; identification of root cause(s); corrective action on the artifact; corrective action on the process), with each change generating another iteration. Examples of activities included in corrective actions: An additional constraint is identified; an assumption is made explicit; a task is formulated to validate an assumption.

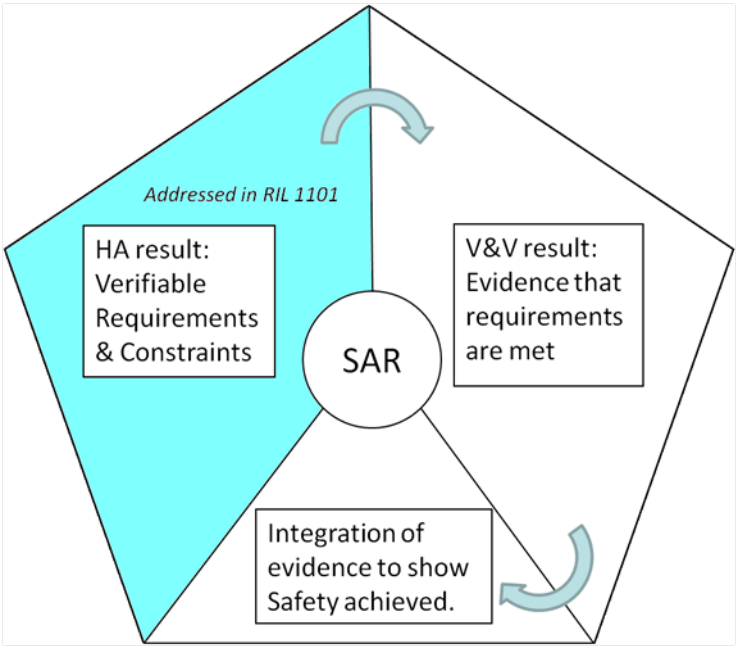


Figure 1: Relationship of HA-evaluation scope in RIL-1101 to overall safety analysis

2.8 Organization of RIL-1101

Section 3 provides the technical basis to support NRC I&C staff in exercising judgment during the review of an applicant's HA. As requested by NRO (the sponsoring User), supporting explanatory information is in the appendices. For example, Appendix C, which is incorporated by reference in Section 3.1 Table 1 item H0-1G, summarizes the state-of-the-art in HA.

Section 3 is organized by groups of contributory hazards, as explained below.

1. These groupings serve as different perspectives on (or projections of) intertwined[19] issues, and are not intended to be mutually exclusive partitions.
2. Subsections 3.1 - 3.3 group contributory hazards that are applicable to all phases of the development lifecycle; typically, these are controlled before starting the development of a particular system.
3. Subsections 3.4 - 3.9 group contributory hazards from the perspectives of individual phases of the development lifecycle.
4. While contributory hazards might manifest themselves or might be discovered in any of several phases of the development lifecycle or levels of integration of a digital safety system, the RIL attempts to place the item in a group corresponding to the earliest prevention opportunity.
5. Relationships between scenarios of contributory hazards (illustrating corresponding hazard spaces) and conditions that reduce these hazard spaces are organized in tables as follows:
   - 5.1. The table title (explained in the narrative introducing it) bounds the scope and context of entries in the rows of the table.
   - 5.2. In a particular row, the left cell includes an example of a scenario[20] illustrating some hazard space.
   - 5.3. A right cell, associated with a contributory hazard in a row, includes an example of a condition that reduces the respective hazard space. Many such conditions could be associated with a particular scenario.
   - 5.4. Each contributory hazard is uniquely identified with a label of the type "H-alpha-<i>"
     - 5.4.1. The "H-alpha-" part of the label is in the column title, applicable to each row, but not repeated. Examples: H-0-; H-culture-; H-OTproc-.
     - 5.4.2. The <i> portion of the label is a numeric, unique to each scenario of contributory hazards.
     - 5.4.3. For example, H-SAE-1 is a complete label for a scenario of contributory hazards.
   - 5.5. A label of the type "H-alpha-<i>-G<j>" identifies a condition G<j> that reduces the H-alpha-<i> space.
     - 5.5.1. For example, H-SAE-1G1 is a condition: associated with H-SAE-1.
6. Hyperlinks are used selectively to identify other salient relationships of the following kinds:
   - 6.1. between scenarios of contributory hazards - possibly across groups (tables)
   - 6.2. between scenarios and conditions reducing the respective hazard spaces
   - 6.3. between conditions reducing the various hazard spaces.

[19] "Many-to-many" interrelationships exist.

[20] In many cases, the scenario is described as a class or category of scenarios.

7. The symbol ↑ as used in the form [H-culture-8↑] in a cell indicates that the item in the cell "contributes to" or is "derived from" the linked item (e.g., H-culture-8).
8. The symbol ↓ as used in the form [H-S-1.1G1↓] in a cell indicates that the item in that cell "requires" the linked item (e.g., H-S-1.1G1).
9. All of the many-to-many relationships are not hyperlinked.
10. Where needed, a note structure, distinguished by indentation, font type and size provides a brief explanation or example for an "H-alpha-<i>" or "H-alpha-<i>-G<j>" paragraph.
11. A link to an item in an appendix leads to further elaboration and background.

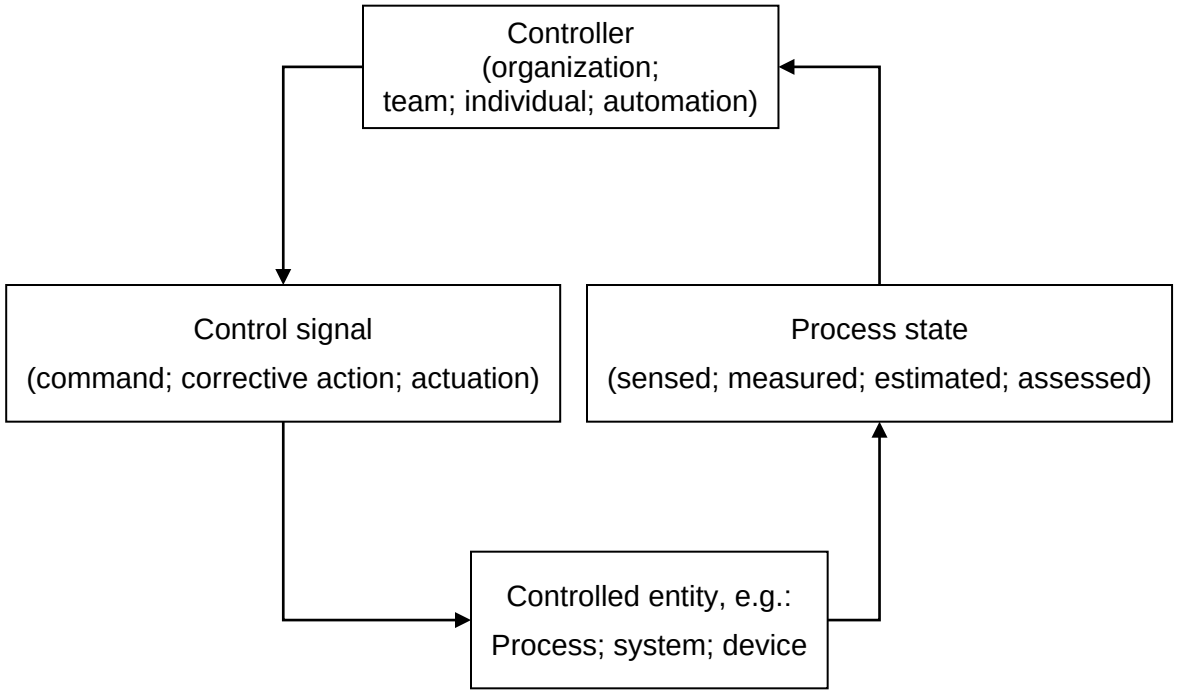
Section 4 explains how HA-review fits in the regulatory framework.

Section 5 summarizes the contribution of RIL-1101 and Section 6 outlines the follow-on research and development (R&D) identified in the course of this work (e.g., unresolved review comments).

Where a word or expression is used in a meaning more specific than or different from the common usage defined in mainstream dictionaries, it is defined in Appendix A: Glossary. Its first occurrence is hyperlinked to that definition.

3 Considerations in evaluating Hazard Analysis

RIL-1101 addresses primarily factors contributing to the degradation of a safety function, rooted in engineering[21]. These factors are part of a network of causes or dependencies (see Appendix K) that result in some defect or deficiency in the system, which could lead to the degradation of a safety function. RIL-1101 refers to these factors as contributory hazards.



[21] rather than random hardware failures during operation

Figure 2: Example of a dependency structure (cyclic graph)

However, recent experience has revealed that propagation paths of hazards are not always linear, and cause-effect relationships are not always direct chains. The indirect propagation of effects (e.g., degradation of a safety function), contributory interactions and propagation paths are not well understood. For example, [11] characterizes these as "*issues that transcend the functions of individual components and involve interactions between components within the system as well as the interaction of the system with the environment.*" Traditional techniques for hazard analysis, as used in common practice, such as fault tree analysis (FTA) [12][13], and failure modes and effects analysis for design (DFMEA) [14][15], do not support the discovery of such contributory hazards well. RIL-1101 is intended to address these gaps.

Experience with complex systems in general [16] and with digital systems for critical functions in diverse application sectors has revealed that common practice does not assure absence of conditions contributing to hazards.

The difficulties NRO experienced (e.g., as reported to ACRS [17]) are examples of the more general trends of increasing system complexity and increasing contribution of systemic causes towards malfunctions. Generally accepted engineering standards[22] do not provide sufficiently specific guidance to ensure their technically consistent, efficient application to digital systems with such complexity. Such reviews require significant additional information [17] from the applicant, significant additional review effort and reliance on judgment, in order to address the gap in the existing review guidance. These gaps were identified in [19] as uncertainties in the assurance of digital safety systems. As depicted in Figure 3, RIL-1101 focuses on the challenges from these uncertainties, characterized as contributory hazards, and identifies corresponding conditions that reduce the respective hazard spaces.

[22] This expression is mentioned in 10 CFR 50.34(a)(ii)(B); examples are referenced in NRC's regulatory guides.

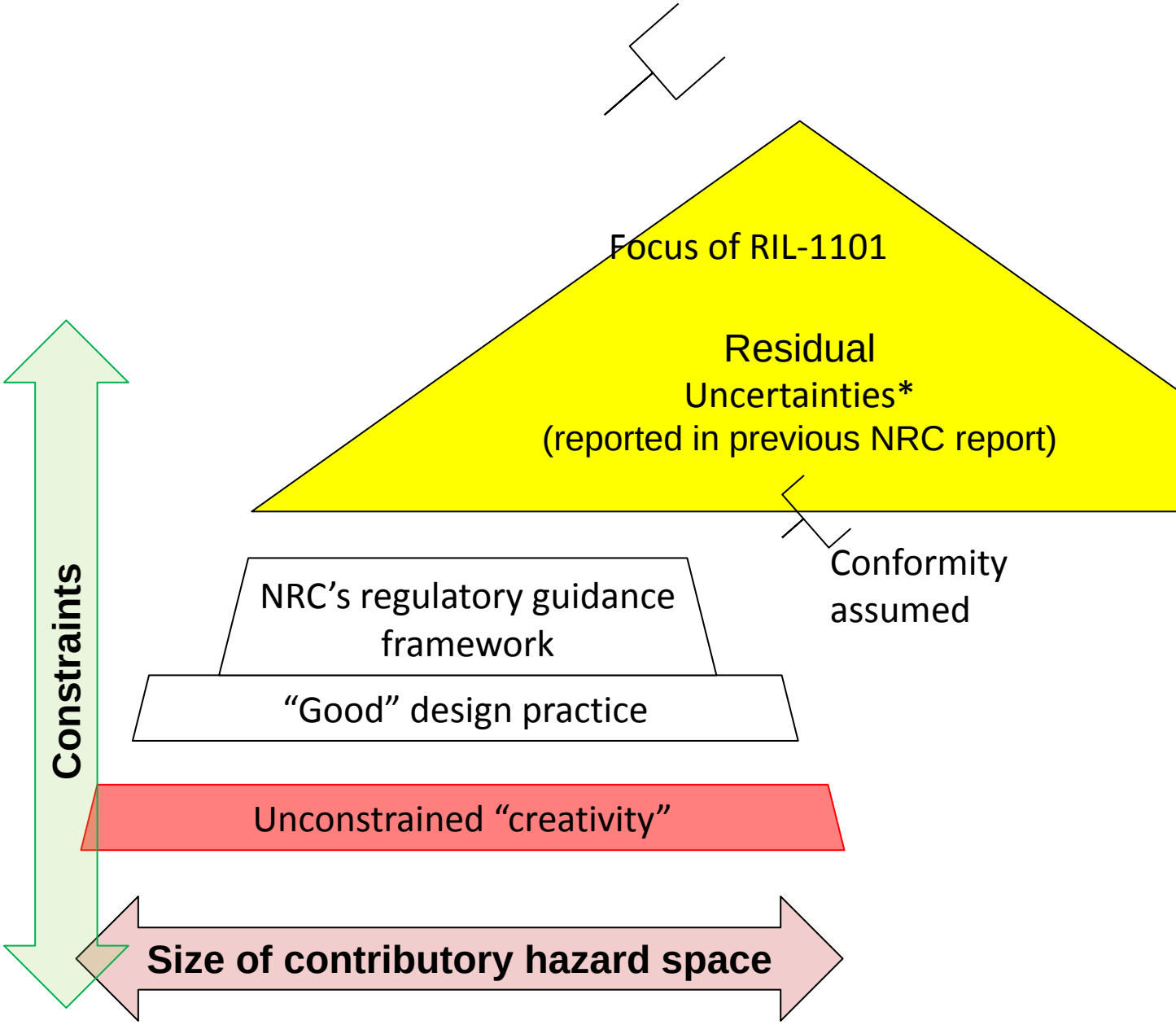


Figure 3: Contributory hazard space in focus

3.1 Evaluation of Overall Hazard Analysis

From the wide range of approaches, methods and techniques to perform hazard analysis, the selection should be well-matched to the object[23] being analyzed. The performers (typically a team) should have the requisite[24] competence. Obscure hazardous conditions are difficult to identify. The corresponding hazard controls should be adequate. The analysis should flow down to all the elements and factors upon which the safety function or its integrity depends. Table 1 includes such overarching considerations in evaluating the HA of a digital safety system. As these factors affect the quality of HA broadly, they are treated as contributory hazards in Table 1. Key considerations are explained in notes after the table and in Appendices C and F.

Table 1: Considerations in broadly evaluating hazard analysis

Contributory hazard		**Examples of conditions that reduce the hazard space**	
ID H-0-	**Description**	**ID H-0-**	**Description**
1	HA approach is not suitable to the system, element, intermediate-phase work product, process or activity being analyzed.	**1G**	The selected HA approach is well-matched to the system aspect, element, development phase, or work product being analyzed, with considerations discussed in Appendix C.
2	Competence in performing HA is not adequate for the system being analyzed. (Also see H-SRE-1)	**2G1**	The HA is performed with the requisite complement of competence; see Appendix C.4 and [H-culture-6G2]. Also see Appendix F.4
3	Validation is inadequate – impaired, because people in the developer's organization are unable to think independently. Intra-organizational reviews suffer from "GroupThink." See Appendix F.4.4	**3G1**	The HA, including elements upon which it is dependent (see: H0-8; H0-9; Appendix K)↓) and the resulting requirements and constraints, is validated (in [21] common position (CP) 2.1.3.2.6) independently, without exacerbating H-culture-9. Also see Appendix F.3. 1. The HA-validation team has the requisite competence [H0-2G1]. 2. The HA-validation team provides perspectives and background different from the team performing the HA.
		3G2	See Appendix F.2 (diversity and independence) and F.4.4 (GroupThink)
6	Hazard controls needed to satisfy system constraints (which prevent hazards) are inadequate	**6G1**	Hazard controls are identified and validated to be correct, complete, and consistent. [H0-7G1↓]
7	Flow down from the controls [H0-6-G1↑] to verifiable requirements and constraints is inadequate.	**7G1**	Requirements and constraints [H0-6G1↑] are formulated and validated to be correct, complete, and consistent in consideration of the preference[25] order 1-4 as follows: 1. Prevent hazard 2. Eliminate hazard 3. Contain hazard (prevent propagation) [H-SR-4G4↓]

[23] For example, techniques in common practice such as FTA, FMEA) may not be very helpful in a situation confounded with interactions and feedback paths.

[24] Proficiency only in FMEA for random hardware failures may not suffice.

[25] It is based on extent of reduction of hazard space, potential fault space, and uncertainty space.

Contributory hazard		Examples of conditions that reduce the hazard space	
ID H-0-	**Description**	**ID H-0-**	**Description**
			4. Monitor, detect and mitigate[26] hazard 4.1. Monitor [H-SR-4G1↓] 4.2. Detect [H-SR-4G2↓] 4.3. Intervene [H-SR-4G3↓] 4.4. Notify (some independent agent)[27] [H-SR-4G5↓] 4.5. (Recipient[28] of the notification) Perform safety-supporting function 4.6. Confirm safe state
8	The analysis is not propagated to elements in an NPP on which the system being analyzed depends or the safety functions allocated to it depend. See in Table 4 H-ProcState-5	**8G1**	All dependencies (see Appendix K) (see: Appendix C Section C.1.1) are identified and analyzed, to confirm that a safety function is not degraded. Also see H-culture-12G2.
9	The analysis is not propagated to processes and process activities on which the integrity of the system being analyzed depends or the safety functions allocated to it depend. See in Table 4 H-ProcState-6↓ H-ProcState-7↓ H-ProcState-8↓	**9-G**	All dependencies are identified and analyzed, to assure that a safety function of the engineered system is not degraded. Processes include organizational processes, management processes, supporting processes, and technical processes. Also see H-culture-12G2.
10	Propagated effect of changes introduces inconsistencies, invalidating previously performed HA.	**10G1**	Starting from the initial HA performed on the functional concept (in [21] CP 2.1.3.2.3) the HA is revised at every phase[29] in the development lifecycle, with change control management and configuration management. Examples of contributory hazards that may be discovered include: 1. Hardware faults 2. Unanalyzed conditions [H-S-1.1.1G1↑].
		10G2	The HA has been iterated until no new hazards are identified [H0_8G1↑]. 1. Added monitoring, detection, mitigation or other requirement has not introduced some new hazard. 2. Some complexity-increasing side effect from the change

[26] Maintain safe state

[27] e.g.: Operator; another automation device or system.

[28] e.g.: Operator; another automation device or system.

[29] Also apply these considerations to successive phases of the system development lifecycle.

Contributory hazard		Examples of conditions that reduce the hazard space	
ID H-0-	**Description**	**ID H-0-**	**Description**
			has not introduced some other, yet-unanalyzed hazard.
10.1	Hazard-introducing effect of iterations is not well understood.	**10.1G**	H0-9.1{G1 – G7}↑ H0-10-G1↑ H0-10-G2↑
11	Required hazard control action is degraded.	**11G1**	Each required control action is analyzed for ways it can lead to the hazard; for example: 1. Not provided; for example: 1.1. Data sent on a communication bus is not delivered. 2. Provided when not needed 3. Incorrect state transition (e.g., combination of 4-5 below). 4. Incorrect value provided; for example: 4.1. Invalid data 4.2. Stale input value is treated inconsistently. 4.3. Undefined type of data 4.4. Incorrect message format 4.5. Incorrect initialization 5. Provided at the wrong time or out of sequence 6. Provided for too long a duration (e.g., for continuous-control functions). 7. Provided for too short a duration; for example: 7.1. Signal is de-activated too early (e.g., for continuous-control functions). 8. Intermittent, when required to be steady; for example: 8.1. Chatter or flutter 8.2. Pulse; spike 8.3. Degradation is erratic 9. Interferes with another action; for example: 9.1. Deprives access to a needed resource; for example: 9.1.1. "Babbling idiot" 9.1.2. Locking up and not releasing resource 9.2. Corrupts needed information 10. Byzantine behavior
12	Hazards in modes of operation other than the "at power" normal mode, or in transition from one mode to another are not adequately understood or analyzed.	**12G1**	HA is performed for all modes of operation (in [21] CP 2.1.3.2.7) and corresponding requirements & constraints are derived (e.g., see checklist in Appendix H).

As HA evaluation progresses further, the selection of information from Sections 3.2-3.9 will be case-specific, depending upon the nature of the object and completeness of product-based analysis.

3.1.1 Considerations for hazards within the system being analyzed

Referring to Table 1, the following notes explain certain contributors to hazards within the system being analyzed:

H0-{6-7; 11}: These factors address the flow down from direct hazards to system constraints to required controls to verifiable requirements and constraints. Sections 3.2-3.9 elaborate on hazard-contributors encountered in the flow down.

H0-{8,9}: Whereas "ineffective hazard recognition" has been recognized as a serious issue [2], unrecognized dependencies (see Appendix C Section C.1.1) are an increasing contributor to this issue, as the complexity of organizations, processes, and systems increases. In addition to the lack of awareness, lapses could occur because of inability to track and maintain a consistent understanding of the dependencies.

H0-8: The extent of dependencies in a system and its elements may not be fully understood or may not be understood in the same way across all parties engaged in developing the system or multiple changes might introduce obscurity. The intent of reviewing for dependencies is to check that the system on which HA is to be performed and its context (environment) are correctly identified, the dependencies correctly understood, conditions that may degrade a safety function (external and internal) are identified, and the commensurate constraints are formulated.

3.1.2 Considerations for hazards contributed through processes

When absence of hazards cannot be ascertained from HA of the system, certain residual uncertainties are addressed by extending HA to the corresponding process-dependencies. When HA has to be extended to processes, a third party certification of the system could provide the requisite confirmation that all process-related dependencies have been identified and their effects analyzed.

H0-9: The extent of dependencies on processes, including the physical processes in the plant, may not be fully understood. For example, Figure 4 depicts an abstraction of process-related direct dependencies. Figure 4 is an example of a generic dependency structure, illustrating how the transformation of a work product depends upon the process activity and factors upon which that activity depends. This process dependency structure can be applied to organize and understand the contribution of organizational processes (Section 3.2), as well as technical processes (Section 3.3). This process dependency structure is also applicable to any other creative, but deterministic, activity, from which predictable, verifiable, analyzable results are needed. Each activity step is affected by the procedures and resources, such as competence (e.g., H-culture-6G2), information, tool, or other aid) employed in performing that activity. The quality of the work product depends upon the quality of the procedures, resources and their utilization, that is, any deficiency is a contributory hazard). Following are examples, indicating less than adequate controls and thus less than adequate understanding of inter-dependencies across processes.

1. Organizational processes lack such controls; or
2. The organization does not apply such controls to the feeder processes or food chain or supply chain; or
3. The organization does not plan for such understanding at the system concept phase of the lifecycle.

H0-10.1: When HA is performed at some stage in the development lifecycle of the system and its elements, additional safety requirements and constraints could be discovered. Inclusion of those requirements[30] could change the system concept or design, requiring another HA cycle to evaluate the impact of such changes. The cumulative and cascading

[30] Incorrect, incomplete, inconsistent, or ambiguous safety requirements can lead to hazards.

effects of these iterations may not be well understood, with the potential to miss subtle implications of a change.

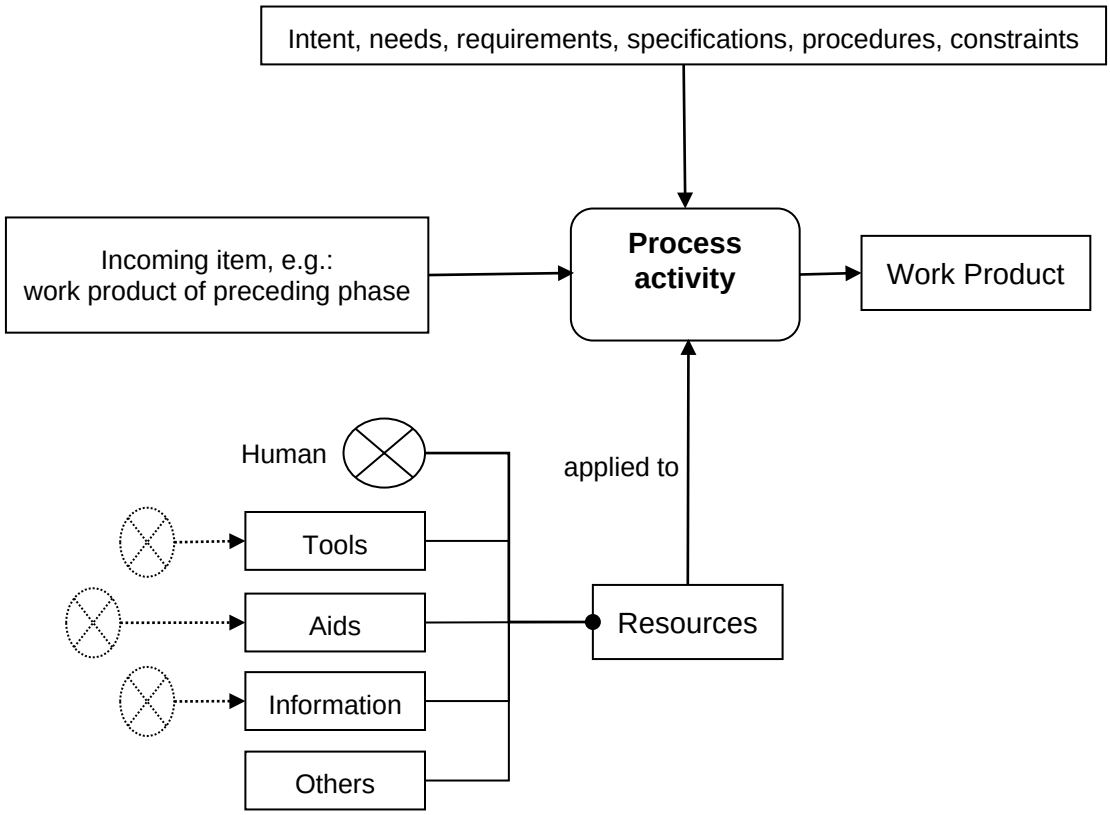


Figure 4: Factors influencing the work product of development

3.2 Evaluation of hazard analysis - organizational processes

Organizational processes include management processes, infrastructural processes, and other supporting processes. The term, “supporting processes” includes change impact analysis process and maintenance processes upon which the system design is predicated.

The culture of an organization with respect to safety engineering and the processes of managing and engineering safety (included within “organizational processes”) have pervasive, permeating effects, that is, the contribution of culture-dependent factors cannot be analyzed[31] as causal events. In software-dependent systems, where the hazard space is much larger than, say, in engineered mechanical structures, these contributors can render the hazards unanalyzable.

[31] This aspect of HA roughly corresponds to but is significantly broader than the HA mentioned in [6] Table 1a.

Table 2 identifies some common concerns.

Table 2: Organization's culture: Examples of contribution to hazards

Contributory hazard		Conditions that reduce the hazard space	
ID H-culture-	**Description**	**ID H-culture-**	**Description**
1	The reward system favors short-term goals, placing cost and schedule over safety and quality (sliding on a slippery slope, not fully cognizant of the cumulative effect of compromises). (Adapted from Annex B in [25])	1G1	The reward system supports and motivates the effective achievement of safety. Safety is the highest priority. Also see Appendix F.3.
		1G2	The reward system penalizes those who take shortcuts that jeopardize safety or quality.
		1G3	The organization has integrity[32].
		1G3.1	The process[33] state is consistent between reality and its representation.
		1G4	Lifecycle economics supporting safety and quality drive the organization.
2	Accountability (e.g., as illustrated in Figure 2 and Figure 4) is not traceable; achievement of safety cannot be assured. Individual accountability becomes lost, because (often without careful reflection) individuals make decisions and evaluate information based on the master premise of the organization. See Appendix F.2.	2G1	The process assures the accountability for effective achievement of safety.
		2G1.1	Influencing factors are organized in an effective control structure[34] (Figure 2), without exacerbating H-culture-9. Also see Appendix F.3.
		2G2	Management commitment to safety motivates effective achievement of safety.
3	Personnel assessing safety, quality, and their governing processes are influenced unduly by those responsible for execution [H-culture-1↑]	3G1	Although information in the processes for safety, quality, verification & validation, and configuration management should be functionally integrated with the main development process to prevent information-loss, the performing personnel are independent (free from undue influence) without exacerbating H-culture-9. Also see Appendix F.3.
4	Personnel feel pressure to conform: 1. "Stacking the deck" when forming review groups. 2. Dissenter is ostracized or labeled as "not a team player" 3. Dissent reflects negatively on performance reviews. 4. "Minority dissenter" is labeled or treated as "troublemaker" or "not a team player" or "whistleblower." 5. Concerned employees fear repercussion. [H-culture-1↑]	4G1	Such behavior is discouraged and penalized. See Appendix F.4.4.
		4G2	The process uses diversity to advantage. 1. Intellectual diversity is sought, valued, and integrated in all processes. 2. "Speaking up" (raising safety concern) is rewarded. 3. See Appendix F.4.2 and F.4.4.
		4G3	Supporting communication and decision-making channels exist and the management encourages their usage (e.g., individual can express safety concern directly to those ultimately responsible). See Appendix F.4.2.

[32] Integrity: Honesty and strength of will to make a safety conscious decision even when it is not popular.

[33] Applicable to any activity in any process in the organization, influenced by its management.

[34] It is a comprehensive safety governance structure, including the higher levels of management.

		4G4	Each identified hazard is logged and tracked to its closure, as explained in Appendix C.3.2-C.3.3. See Appendix F.
4.1	Diminished team ability to seek and use intellectual diversity.	4.1G1	Avoid negative behavior and encourage expression of diverse viewpoints, as explained in Appendix F.4.3.
5	Management reacts only when there is a problem in the field. (Adapted from Annex B in [25])	5G1	Safety and quality issues are discovered and resolved from the earliest stage in the product lifecycle. See Appendix F.
		5G2	The organizational culture has a strongly established master premise of "safety" as the basis for decisions and daily activity. This becomes the guiding premise for analyzing and reducing the hazard space. See Appendix F.
6	The required resources (quality; quantity) are not planned or allocated in a timely manner.	6G1	Resources required[35] are estimated with adequate accuracy[36] in a timely manner.
		6G2	The required resources are allocated in time.
		6G3	Skilled resources have the competence commensurate to the activity assigned. [H0-2G1; H-SRE-1G{1,2,3}]
		6G4	Teams ensure that their knowledge and mental models are properly considered by using communication processes that improve collective mindfulness. See Appendix F.4.
7	A critical cognitive task is interrupted to switch its assignee across multiple tasks; such interruptions could increase the potential of mistakes, thereby increasing the potential fault space or contributory hazard space. (Adapted from Annex B in [25])	7G1	Run critical cognitive tasks to completion (default practice of the organization). Interruption is allowed only when the task has progressed to a stable, well-understood state, such that the interruption does not increase the hazard space.
8	Processes do not produce deterministic, predictable results.	8G1	A defined, documented, disciplined process is followed in all dimensions at all levels, as needed for consistent achievement of safety; for example: 1. Management 2. Engineering 3. Procurement 4. Verification 5. Validation 6. Safety assessment 7. Safety audit
		8G2	The organization follows disciplined communication and cognitive processes to achieve collective mindfulness and know when to adjust and adapt the standardized processes, and learn from the shortcomings. See Appendix F.2 and F.4.
9	When system lifecycle activities are distributed across multiple organizations or parts of the same	9G1	Cross-organizational dependencies are understood clearly.. Also see H-culture-8G2.

[35] Example: Type of competence; degree or level of competence or proficiency; amount of effort time

[36] Implied constraint: Processes are adequately designed and controlled. [H0-9.1G1; H-culture-8G1; H-OTproc-1G]

	organization, safety-relevant information[37] is not communicated efficiently, letting key items of information "fall through the cracks." See note at end of table. [H-SRE-7↓]	9G1.1	The organization maintains cross-organizational connections that improve collective mindfulness; for example, using working groups. See Appendix F.
		9G2	Organizational culture promotes open collaborative communications across boundaries to realize a system that achieves its safety goals. See H-culture-{12G2, 12G3}.
		9G3	Decomposition of safety goals from NPP level analysis and allocation to safety related systems is complete, correct, and consistent and unambiguous.
10	Mistakes repeat.	10G1	Continuous improvement is integral to all processes. See Appendix F.4.
11	Heavy dependence on testing[38] at the end of the product development cycle. By that stage: 1. It often becomes infeasible to correct the problem soundly. 2. Patches increase complexity and impair verifiability.	11G1	H-culture-5G1
		11G2	Technical processes are designed to prevent safety and quality issues as early in the development lifecycle as possible. See Appendix F.
		11G3	Processes for safety, quality, V&V and configuration control are planned[39] and designed to prevent and discover safety and quality issues as early in the development lifecycle as possible. See H-culture-12G2, 12G3} and Appendix F.2.
12	Dependence on implicit information, e.g. implicit assumptions. [H-ProcState-4H0_5↑] [H-OTproc-8↓] [H-SR-11↓]	12G1	All information upon which assurability of safety depends is explicit and configuration controlled.
		12G2	Even while making information explicit and unambiguous, the organization maintains collective mindfulness by persisting in the evaluation of mental models and the development of more accurate and nuanced mental models. This necessarily involves continuous situation awareness of the context and the cultivation of diverse perspectives. See Appendix F.
		12G3	The organization establishes a system for tracking the basis and premise for engineering decisions. See Appendix F.2 and Appendix J.
Note for H-culture-9: Cross-disciplinary, cross-organizational communications quality is affected by stretched lines of communication across the NPP operator (the utility-licensee), the supplier of the plant, the supplier of the DI&C system, and the supplier of components of the DI&C system.			

3.3 Evaluation of hazard analysis - technical processes

Improperly designed or executed technical processes can lead to defects in a system. Examples of technical processes include, but are not limited to the following:

- Requirements engineering – see Section 3.5.
- Architecture engineering – see Section 3.6.

[37] Implied constraint: H0-9G

[38] It is unlikely that testing as the only means of verification will suffice.

[39] Examples of work products: Safety plan; quality plan; V&V plan, demonstrating completeness of coverage.

- Detailed design - see Section 3.8.
- Implementation - see Section 3.9.
- Verification activities by those performing these development activities.
- Third party verification.
- Process assessment.
- Process audit.

Examples of some general contributory hazards and conditions to reduce the respective hazard spaces are given in Table 3 (adapted from Appendix A.1 in [19]), premised on the satisfaction of constraints identified in Table 2.

Table 3: Technical processes: Examples of contribution to hazards

Contributory hazard		**Conditions that reduce the hazard space**	
ID H-OTproc-	**Description**	**ID H-OTproc-**	**Description**
1	Technical processes are not deterministic [H-culture-8↑], that is, correctness of results cannot be assured.	1G	The organization's technical processes are defined to a level of detail such that for each work element involved, there is a specification of the competence, tools, information, and other resources required (see Figure 4) to execute that work element correctly and to integrate results of such work elements correctly. [H-culture-8G1↑]. Also see H-culture-8G2.
2	Any process variable in any work element may contribute to some defect, if not adequately controlled. [H-OTproc-1↑]	2G	Each process variable in each work element is controlled and supported with commensurate methods, tools, and competence to execute that work element correctly and to integrate results of such work elements correctly.. [H-OTproc-1G↑] (Figure 2; Figure 4)
3	Cognitive load (or intellectual complexity) imposed by a specified work element exceeds the capability of assigned personnel. See Note. [H-culture-6↑]	3G1	The cognitive load imposed by a specified work element, including an integration activity, is assured to be well within the capability[40] of personnel available to perform that activity. Also see H-culture-6G4.
3.1	Difficulty of understanding the architecture is a contributor to the cognitive load. Example: Inadequate description.	3G2	The system architecture is analyzable and comprehensible. [H-OTProc-3G1↑]. [H-S-1.1G1↓; H-S-2G6↓]
4	Mistakes (leading to defects) occur[41]; however, technical processes are not designed with the commensurate robustness and resilience to protect from such mistakes.	4G1	The organization's technical processes include processes to detect and recover from mistakes (e.g., verification, audit).
		4G2	H-culture-8G2.
5	The organization believes incorrectly that its processes are adequate, exposing it to unknown sources of defects, for which it	5G1	The process is assessed and certified independently.
		5G2	Qualified independent resources assess the process. [H-culture-6G1; H-culture-6G2]
		5G3	H-culture-8G2.

[40] This may require certification of personnel through a standardized process.

[41] Perfection in human performance is not achievable – at least, not in a sustainable manner.

	cannot identify the causes. [H0-9.1↑; H0-9.3↑]		
6	The processes in real-life execution deviate from the designed processes, resulting in exposure to unknown sources of defects, for which it cannot identify the causes.	6G1	[H-culture-{1G3.1; 2G1.1}
		6G2	The process in execution is audited independently.
		6G3	Qualified resources are available to audit the process.
		6G4	H-culture-6G4 H-culture-8G2. Also see and Appendix F.4.
7	Less accumulated experience and reusable results than in previous generation systems,; for example, shorter lifecycles of implemented systems or configurations leading to • Less accumulated experience on the same item • Changing environments for the same item	7G1	H0-9G H-culture-{2G1.1; 8G1} H-OTproc-{1G; 2G}
		7G2	More rigorous analysis – see Table 1, Table 2. Commensurate conservatively derived requirements and constraints.
		7G3	H-culture-{8G2; 12G2; 12G3}
8	Engineering models lack adequate fidelity to reality, i.e. modeling abstractions are not sound.	8G1	Modeling abstractions are validated.
		8G2	H-culture-8G2
Note for H-OTproc-3: Increasing complexity [16] of systems, processes, and organizations, involving people from multiple organizations, multiple disciplines, multiple locations, and increasing content of software (or other implementation of logic) are increasing the contribution to hazards from engineering activities; for example: • Requirements engineering (elaborated in Section 3.5; HA results in safety requirements & constraints). • Architecture engineering (elaborated in Section 3.6) • Software engineering, elaborated in Sections 3.6.4 and 3.8			

3.4 Evaluation of Hazard Analysis - System Concept

The system concept, sometimes known as the functional concept (of the intended system), is described in terms of the initial requirements associated with it and its relationship with its environment, including the boundary and the assumptions (see Appendix J) on which these are based. Sometimes, the associated requirements are embodied in a "concept of operations" document. Sometimes HA[42] of a functional concept is called preliminary hazard analysis (PHA[43]) – (also see Appendix C-2.

In practice, the degree of specificity of a system concept varies over a wide range; sometimes the initial concept is so vague that it leads to misunderstandings, lapses, or inconsistencies, which contribute to hazards. Application and evaluation of HA (Section 3.1) is most effective in the concept phase of a system development lifecycle. Avoidance of these contributors to hazards (see Table 1; Table 21 tasks T1-T3.) requires clear description and tracking of the evolving system concept and its relationship with its environment, as discussed in this section.

3.4.1 Hazards associated with the environment of the DI&C system

Hazards can be contributed through an ill-understood relationship between the conceived system and its environment, some examples of which are given in Table 4, Table 6, and Table 7. These tables also identify conditions that reduce the respective hazard spaces.

[42] It roughly corresponds to but is significantly broader than the HA mentioned in [6] Table 1b.

[43] These are good candidates for discussion with the applicant before it submits the license application.

Hazards (including contributory hazards) may originate in the environment of the analyzed DI&C system, or may originate in the DI&C system, or may result from their interactions. See Appendix E.4 for hazard sources from the physical environment. See Appendix E.5 for ways in which a DI&C system may affect its environment adversely.

Section 3.4.1.1 includes examples of hazards related to interactions with the plant processes.

Section 3.4.1.2 includes examples of hazards related to interactions with instruments, controls, and networks in the system's environment.

Section 3.4.1.3 includes examples of hazards contributed through human-interaction aspect of the system's environment.

Section 3.4.2 includes examples of hazards contributed through deficiencies in the architectural concept. Conditions reducing the hazard space are applicable recursively to architecture inside the intended safety system in every phase in the development lifecycle (from conception to implementation), to every level in the system architecture integration hierarchy, and to transformations from one level to another.

3.4.1.1 Hazards related to interaction with plant

Often, hazards arise from an inconsistency between the perceived process state and the real process state. Here, the term "process state" is used in the general sense, for example: the state of the nuclear reaction process, the state of some supporting physical process in the NPP, the state of control automation, the state of some instrument, or even the degradation process of some device. Hazards can also arise from unanalyzed conditions in the joint behavior of the plant (including equipment and processes) and the safety system. Table 4 shows examples of contributory hazards and conditions that reduce the respective hazard space.

Table 4: Interaction with plant: Examples of contribution to hazards

Contributory hazard		**Conditions that reduce the hazard space**	
ID H-Proc State -	**Description**	**ID H- Proc State -**	**Description**
1	The nature of change in some monitored physical phenomenon[44] in the process of interest in the environment of the digital safety system is not well understood or not characterized correctly. Also see H-SR-23	1G1	The physical processes[45] in the monitored phenomenon are modeled and represented correctly; for example:
		1G1.1	• Nature of variation over time
		1G1.2	• Dependencies on other phenomena
		1G2	The perceived state matches reality with the fidelity required in value and time.
1.1	The temporal aspect of change in a continuously varying phenomenon is not well understood or not characterized correctly.	1.1G1	Temporal behavior of a continuously varying phenomenon is characterized correctly. such that timing requirements for monitoring it can be derived without loss of fidelity. This includes timing relationships across monitored phenomena,
		1.1G1.1	The physics of the phenomenon (e.g., dynamic behavior,

[44] Examples: Pressure; temperature; flow; neutron flux density

[45] Examples: Energy-conversion; equipment degradation; component degradation

			including disturbances) is understood well and characterized mathematically.
1.2	The temporal aspect of change in a sporadic phenomenon is not well understood or not characterized correctly.	1.2G1	Requirements for reacting to sporadic events (e.g., sudden change) include the minimum inter-event arrival time, based on the physics of the event-generating process.
		1.2G2	Signal indicating event of interest is not filtered out.
		1.2G3	Signal indicating event of interest is not missed due to inadequate sampling, as determined through mathematical analysis.
		1.2G4	Capturing event of interest does not disrupt any other action upon which a safety function depends.
2	Unanalyzed joint behavior of the safety system and the plant equipment and processes degrades a safety function.	2G1	Safety system and its environment, including the NPP equipment and processes are analyzed as a coupled system with sufficiently deep models of the behaviors (e.g.: processes; instruments; controls; networks) to represent reality with fidelity[46].
3	Allocation of safety functions and properties from a system at a higher level of integration to one at a lower level, is not correct, complete or consistent, or is ambiguous. See notes.	3G1	Relationships with losses of concern identified at NPP level analysis and commensurate safety goals formulated in NPP level analysis are explicit.
		3G2	Decomposition of safety goals into required safety functions (design bases) is complete, correct, and consistent and unambiguous.
		3G3	Allocation of safety requirements to safety related systems[47]) is complete, correct, consistent and unambiguous. Also see Table 9.
		3G4	Allocation of safety properties, including corresponding decomposition or flow-down or derivation of constraints, is complete, correct, and consistent. See Section 3.5.1.1. Table 8.
		3G5	The boundary of the system being analyzed is well-defined with respect to its environment (in [19] CP 2.1.3.2.1).
		3G6	Interface to and interactions with the plant are specified and constrained in a manner that the system is understandable [H-S-2↑], verifiable[48] [H-S-1.1], and free from interference [H-S-3]). Examples of elements in the environment include interfaces to and interactions with: 1. Sensors 2. Actuators 3. Services needed; for example: 3.1. Electricity 3.2. Air flow 3.3. Compressed air 3.4. Water 4. Human-machine interfaces 4.1. Roles, responsibilities, functions. 4.2. Procedures specifying 4.1.

[46] Traditional FMEA and FTA of I&C systems in the plant will not suffice, as noted elsewhere.

[47] If there are multiple levels of assembly (integration) this criterion applies to each level-pair.

[48] i.e., satisfaction of the constraint or specification is verifiable by analyzing the system concept.

		3G7	Constraints on other elements in the environment of the system are explicit. Restrictions & constraints placed on the system are explicit; example constraints: 1. Compatibility with existing systems. 2. Physical and natural environment. 3. Protection against propagation of non-safety system faults and failures.
		3G8	Restrictions & constraints placed on the system are explicit; example constraints: 4. Compatibility with existing systems. 5. Physical and natural environment. 6. Protection against propagation of non-safety system faults and failures.
4	Interactions of the system with its environment, including effects of assumptions, are not well-understood. [H-ProcState-3↑] See note. (In [19] Appendix A.3 item 3). [H-culture-12↓]	4G1	See: H-Pr**oc**State-3G7; H-culture-{12G2, 12G3}; Appendix J.
		4G1.1	[H-culture-12G1↓] The organizational processes (Section 3.2) include explicit tasks or activities to validate each assumption in time to avoid adverse impact on the system safety properties and HA activities. Also see H-culture-{12G2, 12G3}.
		4G1.2	If an assumption is found to be invalid or there is a change from the previous assumption: 1. There is a corresponding change impact analysis, maintained as an independently evaluated configuration item. 2. The affected part of the HA is repeated 3. Commensurate changes in constraints or requirements are identified. 4. There is an analysis of the impact of those changes. 5. The change impact analysis is an independently evaluated configuration item.
		4G2	Hazards from the physical environment are analyzed. See Appendix E.4
		4G3	Hazards from the DI&C system on its environment are analyzed. See Appendix E.5
Note for **H-ProcState-{3-4}:** The intent of reviewing for these factors is to check that the system on which HA is to be performed and its context (environment) are correctly identified, the dependencies are correctly understood, the primary hazards (external and internal) are identified, and the commensurate constraints are identified.			
Note for **H-ProcState-3**: When a large complex system, such as an NPP (including its environment and processes for operation and maintenance) is decomposed into manageable subsystems and components, the constraints necessary to prevent the losses at the top level (e.g., NPP-level) may become obscure. For example, subtle couplings across the decomposed elements might arise. In an evolving configuration of the overall (e.g., NPP-level) system, the boundary of the system being analyzed and assumptions (see Appendix J) about its environment may not be well-defined, leading to appropriate considerations "falling through the cracks."			

Figure 5 depicts a "progressive[49]" migration from normal operational process state region (shown in green) to an unsafe state region (shown in red). Actions to avoid the unsafe state region (i.e. to effect safe recovery) need some time (shown as the brown region). To allow for the needed time, the temporal aspect of change in the monitored phenomena must be understood well and departure (shown in yellow) from normal operational state, monitored. Intervention must be completed within this (yellow) region.

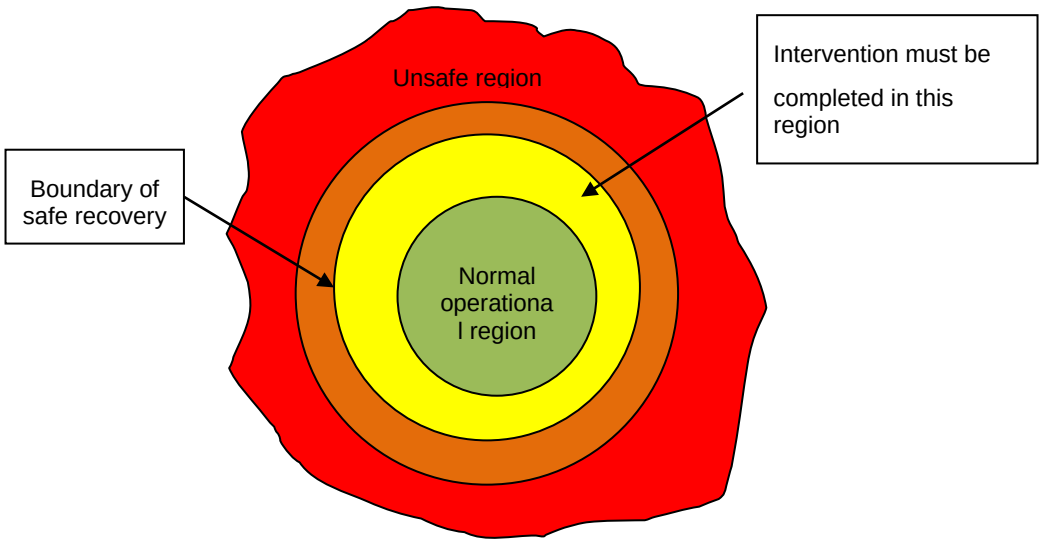


Figure 5: Regions of state space for hazard analysis

[49] Premise: Degradation is not sudden or unpredictable, and progression can be monitored.

Table 5: Interdependencies: Examples of contribution to hazards

Contributory hazard		Conditions that reduce the hazard space	
ID H-Dep-	**Description**	**ID H-Dep-**	**Description**
1	**Unrecognized inter-dependencies in the system:** Inter-dependencies in the system, its elements, and its environment (see H-ProcState-4) are not understood, recognized or explicitly identified, leaving some vulnerability, which can lead to the degradation of a safety function. [H0_8↑]	1G1	All inter-dependent systems, elements, processes, and factors affecting a safety function are identified. See H-culture-{8G2, 9G2}.
		1G1.1	Design rationale is recorded and tracked. See Appendix F.2.
		1G2	The items identified in are configuration items.
		1G3	The inter-dependencies or relationships among these items are unambiguously described, especially those affecting emergent behavior. [**H-ProcState**-5G1↑] Also see H-culture-{12G2, 12G3}.
		1G4	Semantics of the relationships are explicit: Relationships may not merely be sequential (chained) or tree-structures, but also cycles – often feedback control loops[50]. [**H-ProcState**-5G1↑]
		1G5	The inter-relationships of these configuration items are identified (e.g., by means of an overall NPP-level architecture). [**H-ProcState**-5G1↑]
		1G6	These inter-relationships are also a configuration item or set of configuration items. [H-ProcState-5G5↑]
		1G7	Independent verification assures that these configuration items represent reality. [**H0**-8.1G1↑]
		1G8	Effect of these dependencies is analyzed to prove that the safety function is not degraded.
		1G9	Any change in any of these configuration items is managed through a change control process, with a documented analysis of the impact of change. (Generalized from CP 2.7.3.1.5 in [21]) [**H-ProcState**-5G1↑] See Appendix F.2.
		1G10	The change impact analysis is independently verified. [H-ProcState-5G8↑]
		1G11	The change impact analysis is a configuration item. [H-ProcState-5G8↑]
1.1	Dependencies through the environment of the digital safety system are not recognized; for example: • The physical processes • Degraded behavior of related instrumentation and peripheral equipment	1.1G1	Effect of these dependencies is analyzed to prove that the safety function is not degraded.
		1.1G2	H-culture-8G2
2	**Unrecognized inter-dependencies in the development process:** Inter-dependencies in the system development process, feeder	2G1	All inter-dependent processes (including feeder and supporting processes), resources used in these processes and factors affecting these processes and resources are identified (e.g., see Figure 4). See H-culture-{8G2, 9G2}.

[50] Contrast with a chain of events initiated by failure of a hardware component

	processes, supporting processes, elements, and environments, are not understood, leaving some vulnerability, which can lead to a defect in the system, which could lead to the degradation of a safety function. [H-0-9↑]	2G2	These are configuration controlled items (henceforth, configuration items). [H-ProcState-6G1↑]
		2G3	The inter-dependencies or relationships among these items are unambiguously described, including cycles created through feedback loops[51]. [H-ProcState-6G1↑] Also see H-culture-{12G2, 12G3}.
		2G4	The inter-relationships across these configuration items are identified (e.g., by means of an overall process architecture), and are also a configuration item or set of configuration items. [H-ProcState-6G1↑]
		2G5	Some combination of independent assessment, audit, and verification assures that these configuration items represent reality. [H-ProcState-6G1↑]
		2G6	Any change in any of these configuration items is managed through a change control process. [**H**-ProcState-6G1↑]
		2G7	Effect of these dependencies is analyzed to prove that the safety function is not degraded.
		2G8	H-culture-8G2
3	Dependencies through supporting services and processes are not recognized	3G1	Effect of these dependencies is analyzed to prove that the safety function is not degraded. See H-culture-{8G2, 9G2}.
		3G2	H-culture-8G2
4	Dependencies through resource[52] sharing are not recognized; examples: • Contention for the shared resource • Corruption of resource (e.g., data)	4G1	Effect of resource-sharing is analyzed to prove that the safety function is not degraded. See H-culture-{8G2, 9G2}.
Note: Whereas "ineffective hazard recognition" has been recognized as a serious issue [1], unrecognized dependencies are an increasing contributor to this issue, as the complexity of organizations, processes, and systems is increasing. In addition to the lack of awareness, lapses could occur because of inability to track and maintain a consistent understanding of the dependencies. The state of practice in representing and analyzing such dependencies is relatively weak, as discussed in Appendix C.			

3.4.1.2 Contributory hazards from NPP-wide I&C architecture

The scope of NPP-wide system architecture includes the safety system under evaluation and its relationship with its environment, that is, all systems, elements, processes and conditions that support or affect the performance of a safety function. "Relationship" includes interfaces, interconnections, and interactions, whether these are direct, intended, explicit, static, "normal," indirect, implicit, unintended, dynamic, or "abnormal." Any relationship that affects the performance of a safety function is a dependency. HA of the NPP-wide I&C architecture should examine it for hazards relevant to the safety related system to be analyzed. Figure 6 provides a simplified view.

[51] These can also be analyzed as control loops influencing safety properties of the affected system.

[52] Examples: Skilled resources for development; Computing memory or processor-time during execution.

Constraints on the NPP-wide I&C architecture are derived from the quality[53] attributes or properties of the safety related system being analyzed. Quality attributes are discussed in Section 3.5.1.1, including Table 8, which also applies to the NPP-wide I&C architecture.

> Note: Criteria for the HA-evaluation the NPP-wide architecture are predicated on the correct and complete performance of HA, as illustrated in Table 1, including considerations of combinations of multiple contributory hazards, exemplified through Table 4, Table 2, Table 3, Table 6, and Table 7.

Table 13, derived from considerations in Table 8, also applies to the NPP-wide I&C architecture. in the context of hazards contributed through interference.

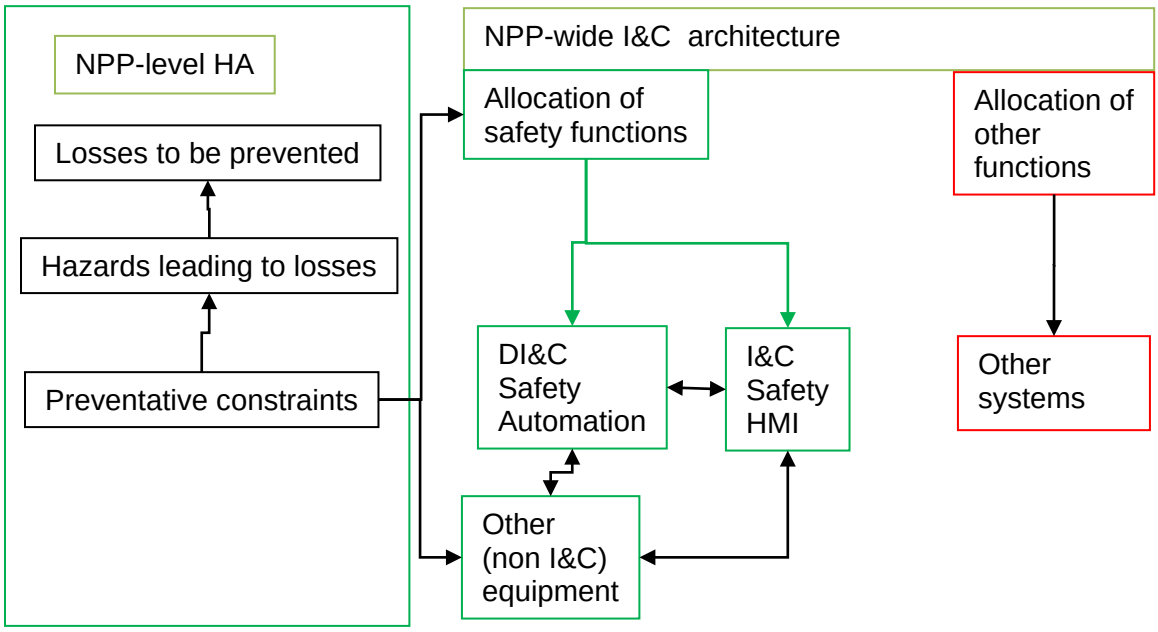


Figure 6: NPP-wide I&C architecture - allocation of functions in concept phase

[53] Other terms for these properties: Quality-of-service (QoS) properties; non-functional requirements

3.4.1.3 Contributory hazards from human machine interactions

Hazards of the kind grouped in Table 1 - Table 4 could also affect human-automation interactions

Table 6 supplement those with some examples of more specific hazards contributed through human-automation interactions and Table 7, those through inadequacies in the associated engineering.

Table 6: Human machine interactions: Examples of Contribution to hazards

Contributory hazard		**Conditions that reduce the hazard space**	
ID H-hmi-	**Description (**e.g., Scenario**)**	**ID H-hmi-**	**Description**
1	Inconsistency between human-perceived process state and real process state	1G1	Process state presented to the human represents the real physical state in value and time.
2	Inconsistency between human-perceived state of an instrument[54] and real state of the instrument	2G1	Instrument (e.g., actuator) state presented to the human represents the real physical state of the instrument in value and time.
3	Mode confusion	3G1	Human is notified of the current mode and a mode change in progress (the loop is closed with feedback).
		3G2	Human has a correct understanding of the mode-change model (human is equipped with correct mental model of the mode-switching behavior of the automation)
		3G3	Potential for mistaken interpretation of the information presented by the human-machine interface is eliminated.
		3G4	Inconsistent behavior of automation is avoided; or, automation detects its inconsistency and notifies human.
		3G5	Unintended[55] side effects are avoided
3.1	Confusion about line of authority (who or what entity is in control at the moment)	3.1G1	Multiple concurrently active paths of control authority (logical control flow) are avoided
		3.1G2	Change of mode by automation without human confirmation is avoided.
		3.1G3	Correct division of tasks is ensured through analysis of human tasks, including human-automation interactions.

[54] Example: A sensor; an actuator.

[55] Any intended effect is explicit (e.g., as a part of the specification) and analyzed for its effect on a safety function.

4	Inappropriate division and allocation of tasks between human and automation.	4G1	H-OTproc-3G1
5	Normally useful cognitive processes are defeated or fooled by a particular combination of conditions [11]	5G1	See H-hmi-6G1
6	Human mental model of how the system works is not consistent with the reality.	6G1	"How the system works" (the information needed by operating personnel about its behavior and needed automation -human interaction) is described clearly, including behavior and automation-human interaction under off-normal conditions (e.g.: presence of a fault; all combinations of such conditions).

Table 7: Human machine interaction engineering: Examples of Contribution to hazards

Contributory hazard		**Conditions that reduce the hazard space**	
ID H-hmiP-	**Description (**e.g., Scenario**)**	**ID H-hmiP-**	**Description**
1	Loss of information across disciplines (e.g., automation engineering, human factors engineering, control room design). [H-culture-9↑][H-SR-3↑]	1G1	System is engineered holistically, including crosscutting analysis. (Adapted from [19] Appendix A.3 footnote 82)
2	Confusing human-machine interface design	2G2	H-hmi-3G3
3	Cognitive overload	3G3	H-OTproc-3G1

3.4.2 Contributory hazards in conceptual architecture

The term "conceptual architecture" refers to the architecture of the system concept, as it evolves in relation to its environment (also see Sections 3.4.1.2).

Here, the focus shifts from the interactions of the conceived system with the environment to its internal architecture, as driven by the requirements allocated to it, that is, the inter-relationships of the various requirements and constraints to be satisfied by the conceived system. The information in Table 8 and Table 13 is applicable to the conceptual architecture, especially with respect to the following concerns:

1. Freedom from interference across redundant divisions [Table 13 H-S-3G3 - 2↑].
2. Freedom from interference between a monitoring element and its monitored element [Table 13 H-S-3G3 - 4↑].

3. Compromise of redundancy through a dependency (e.g., input data; resource-sharing). Also see Table 1 item 8-10.
4. Compromise of redundancy in the concept of voting[56] logic.

The conditions (to reduce the respective hazard spaces) provided in Table 8 and Table 13 apply recursively to the finest grain level of the system architecture and recursively to the finest grain level of the software architecture. These conditions also apply to the mappings (e.g., through composition-decomposition) from one level to another in the architecture hierarchy[57] and through all stages of derivation of requirements & constraints and the subsequent development lifecycle stages (e.g., detailed design and implementation).

3.4.3 Contributory hazards from conceptualization processes

Examples of hazards contributed through weaknesses in the cultural and general technical processes of the organization (Table 2 and Table 3), which were introduced in Section 3.2, apply to the concept phase of the system development lifecycle strongly.

Requirements engineering (Section 3.5) and architectural engineering (Section 3.6) apply to the concept phase also – see Table 12, Table 13, and Table 14.

Planning the rest of the development lifecycle goes hand in hand with the conceptualization, as stated in Appendix C.3 Table 21, tasks T1-T3.

3.5 Evaluation of hazard analysis - Requirements

Identifying valid requirements for the digital safety system has been found to be one of the weakest links in the overall process, in the experiences of many critical application domains. Inadequacy in requirements is one of the most common causes of a system failing to meet expectations. Failures traceable to shortcomings in requirements cannot be caught through such verification activities as simulation and testing alone. Formal methods do not help in understanding intent or eliciting missing requirements, when the intent is not clear [19]. For a safety system, requirements and constraints emerge from hazard analysis and are validated through independent hazard analysis. Although initial requirements for a digital safety system come from a higher level of integration (e.g., from a NPP-level safety analysis), additional requirements and constraints are discovered at every phase of the development lifecycle.

3.5.1 System Requirements

In the general context of systems engineering, the specification of a primary function, valued and required by its user, is called a functional requirement. In the context of digital safety systems, example groups of functional requirements include (but are not limited to) monitoring departure from a safe state, detecting threshold for intervention, and intervention for mitigating the consequence of departure from safe state. Key prerequisite activities for identifying safety requirements were discussed in Sections 3.1 (overall hazard analysis, understanding dependencies leading to loss events), 3.4.1 (understanding hazards in relation to the environment of the safety system), including hazards contributed from inadequate definition of the boundary of the safety system, invalid assumptions (see Appendix J), and interactions with

[56] Example: In a quad-redundant system for a space system, four computers were connected by a multiplexor/de-multiplexor module. A diode in the interconnections failed in an unanticipated way, such that the condition was not observed by the 4 computers similarly. (In [19] Appendix A.3 footnote 84)

[57] The mapping could contribute a hazard, e.g., Some abstractions can mask problems.

other systems and humans). The analysis reviewed in those sections contributes to an early stage of requirements engineering. Given the requirements resulting from those analytical activities, Section 3.5.1.1 introduces the concept of associated quality requirements. Section 3.5.1.1 also introduces the concept of derived quality characteristics or requirements in an organizing framework, known as a "quality model" [30]. Section 3.5.1.2 identifies some common weaknesses in formulating verifiable requirements, and Section 3.5.1.3 identifies some common weaknesses in the associated requirements engineering processes.

3.5.1.1 Quality requirements

Figure 7 shows quality requirements associated with functional requirements. In the context of this RIL, examples of top-level quality requirements are Safety and Security.

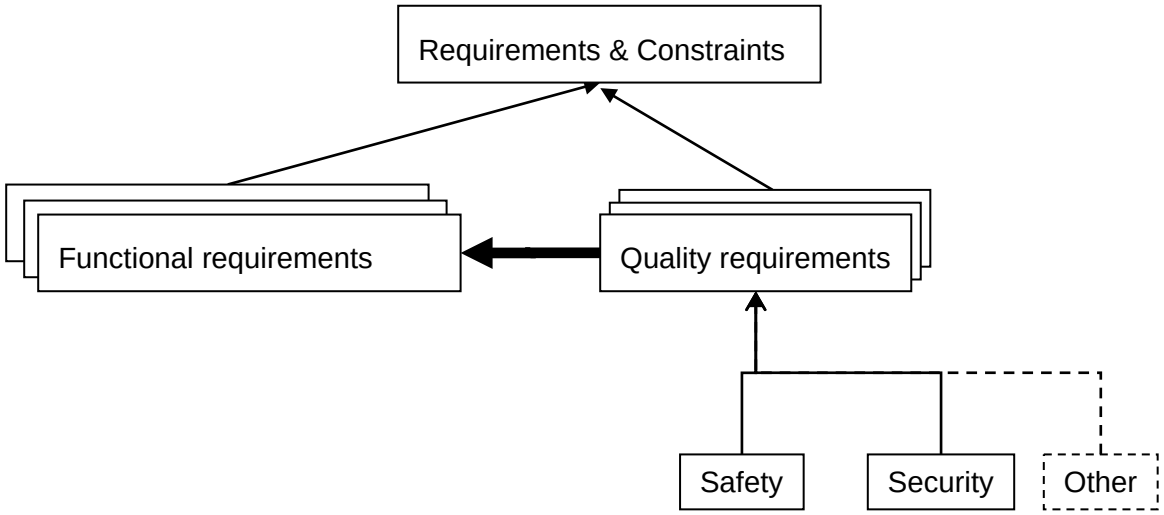


Figure 7: Quality requirements should be explicit

For a safety system[58], as shown in Figure 8, the "Assurability" property distinguishes it from systems that do not require similar assurance[59]. Figure 8 also shows other quality attributes that contribute to or support "Assurability." The corresponding quality requirements may also be viewed as constraints to be satisfied by the digital safety system, that is, constraints on the solution space (also known as design space), such that system concepts that do not satisfy these constraints are eliminated from further consideration (i.e., the hazard space is reduced). Table 8 shows the logical derivation of these constraints (with the derivation relationships shown in Figure 8) to support the "Assurability" property with the following informally expressed reasoning:

1. To be able to assure that a system is safe, one must be able to verify [H-S-1] that it meets all its safety requirements.
2. For a system to be verifiable, it must not be possible for one element of the system to interfere with another. [H-S-3]

[58] As stated in Section 2.3.3, the scope is limited to the automation. People are part of its environment.

[59] Example: A commercial-grade system or element on which no safety function is dependent.

3. If the conceived system is too complex, adequate verification is infeasible. [H-S-1.1]
4. If one cannot even understand it, how can one assure that it is safe? [H-S-2]
5. Verifiability is a required system property, flowing down from the system to its elements (constituents) progressing to the finest-grained element; it implies corresponding verifiable specifications. Verification also includes analysis at various phases in the development lifecycle, well before[60] an artifact is available for physical testing. Examples of conditions for verifiability:
   5.1. Ability to create a test (or verification) case to verify the requirement.
   5.2. Observability
   5.3. Ability to constrain the environment of the object of verification.
6. For "analyzability" the system must have predictable and deterministic[61] behavior. [H-S-1.2].

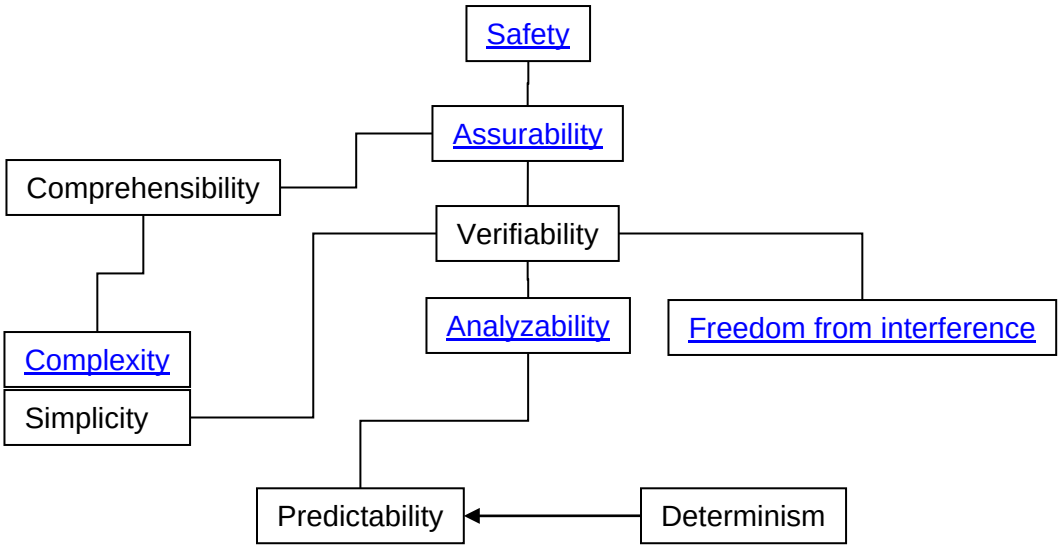


Figure 8: Quality characteristics to support safety

Table 8: Constraints derived from quality attributes: Scenario-based examples

Contributory hazard		**Conditions that reduce the hazard space**	
ID H-S-	**Generalized Scenario**	**ID H-S-**	**Description**
1	The system is not sufficiently verifiable and understandable, but this deficiency is discovered too late. Appropriate considerations and criteria are not formulated at the beginning of the development lifecycle; therefore, corresponding architectural constraints are not formalized and checked. When work products are available for testing, it is discovered that adequate testing is not feasible (e.g., the duration, effort, and cost are beyond the	1G1	Verifiability is a required system property, flowing down from the system to its constituents progressing to the finest-grained element. (Adapted from CP 2.2.3.11 in [21]) [H-S-1.1G1↓]
		1G1.1	Verifiability of a work product is checked at every phase of the development lifecycle, at every level of

[60] When performed on a computer program (code), it is known as static analysis. However, analysis in the same "static" sense can also be performed on work products of earlier phases, e.g. on models. [H-S-1.1.1]

[61] Yields deterministic results.

Contributory hazard		Conditions that reduce the hazard space	
ID H-S-	**Generalized Scenario**	**ID H-S-**	**Description**
	project limitations).		integration, before proceeding further in the development.
1.1	System is not verifiable (e.g., it is not analyzable or very difficult to analyze).	1.1G1	Avoidance of unnecessary[62] complexity
		1.1G1.1	The behavior is unambiguously specified for every combination of inputs (including unexpected inputs) at every level of integration in the system (in [19] Appendix A.4 item 4).
		1.1G1.2	The flow-down ensures that 1. Allocated behaviors satisfy the behavior specified at the next higher level of integration; 2. Unspecified behavior does not occur.
		1.1G1.3	The behavior of the system is a composition of the behaviors of its elements, such that when all the elements are verified individually, their compositions may also be considered verified[63]. This property is satisfied at each level of integration, flowing down to the finest-grained element in the system.
		1.1G1.4	Development follows a refinement process.
1.1.1	There are unanalyzed or un-analyzable conditions. For example, all system states, including unwanted ones such as fault states, are not known and not explicit. To that extent, verification and validation (V&V) of the system is infeasible. [H-S-1.1↑]	1.1.1G1	Static analyzability: System is statically analyzable. 1. All states, including fault conditions, are known. 2. All fault states, leading to failure modes, are known (in [21]CP 2.2.3.14 1st item). 3. Safe state space of the system is known (in [21] CP 2.2.3.14 2nd item).
1.1.2	There is inadequate evidence of verifiability. [H-S-1.1↑]	1.1.2G1	Verification plan shows the coverage needed for safety assurance.
1.2	System behavior is not deterministic[64]. [H-S-1.1.1↑]	1.2G1	System has a defined initial state.
		1.2G2	System is always in a known configuration.
		1.2G3	System is in a known state at all times (e.g., through positive[65] monitoring and indication): 1. Initiation of function

[62] Meaning: [Complexity] that is not essential to support a safety function.

[63] No unspecified behavior emerges.

[64] Yields deterministic results.

[65] If indirect indication or inference is used, HA confirms satisfaction of H-ProcState-1G1.2.

Contributory hazard		Conditions that reduce the hazard space	
ID H-S-	**Generalized Scenario**	**ID H-S-**	**Description**
			2. Completion of function (in [21] CP 2.1.3.4 last item) 3. Intermediate state, where needed to maintain safe state in case of malfunction.
1.3	System behavior is not predictable. [H-S-1.1.1↑]	1.3G1	Each transition from a current state (including initial state) to some next state is specified and known, including transitions corresponding to unexpected combination of inputs and transition conditions.
		1.3G2	A hazardous condition can be detected in time to maintain the system in a safe state. (in [21] CP 2.2.3.14 3rd item).
2	Comprehensibility: System behavior is not understood completely and correctly by its community of users (e.g., reviewers, architects, designers, and implementers), that is, the people and the tools they use. [H-S-1↑]	2G1	Behavior is completely and explicitly specified. Also see H-culture-{12G2, 12G3}.
		2G2	Behavior is completely understandable. Also see H-culture-{12G2, 12G3}.
		2G3	Behavior is understood completely, correctly, consistently, and unambiguously by different users interacting with the system. Also see H-culture-{12G2, 12G3}.
		2G4	The allocation of requirements to some function and that function to some element of the system is bi-directionally[66] traceable. (in [19] Appendix A.4 item 2).
		2G5	The behavior specification avoids mode confusion, esp. when functionality is nested (in [19] Appendix A.4 item 3).
		2G6	The architecture is specified in a manner (e.g., language; structure) that is unambiguously comprehensible to the community of its users (e.g., reviewers, architects, designers, implementers), that is, the people and the tools they use (in [19] Appendix A.4 item 9).

Considering that the state of practice is especially weak in the derivation of verifiable constraints from quality requirements, a careful review is needed. The architecture should satisfy these constraints, starting from the system concept phase and continuing at every successive phase of development, refinement and decomposition, including all phases of the software development lifecycle. Commensurate architectural constraints are identified in Section 3.6.

[66] It is not implied that one-to-one relationships are necessary.

3.5.1.2 Contributory hazards through inadequate system requirements

Activities leading to identification of functional requirements for safety were introduced in Sections 3.1 (overall hazard analysis, including understanding dependencies leading to a loss event or degradation of a safety function), 3.4.1 (understanding hazards in relation to the environment of the safety system), including hazards contributed from inadequate definition of the boundary of the safety system, invalid assumptions (see Appendix J), and interactions with other systems and people). Table 9 identifies further contributory hazards due to weaknesses in identifying and formulating requirements. The content of Table 9 is adapted mostly from Appendix A.3 in [19]; other sources are referenced within the respective item in Table 9. For hazards contributed through weaknesses in interfaces and interactions across elements of the system, see Section 3.6.1.

Table 9: Inadequacy in system requirements: Examples of contribution to hazards

Contributory hazard			**Conditions that reduce the hazard space**
ID H-SR-	**Description (**e.g., Scenario**)**	**ID H-SR-**	**Description**
1	Mistakes occur due to misunderstanding the environment	1G1	[H-SRE-{1G1, 1G2, 1G3}↓] See: H-culture-{4G1, 4G2, 4G3, 6G3}; See Appendix C.3.2-C.3.3; Appendix F, esp. Appendix F.1
2	Input constraints misunderstood or improperly captured [H-SR-1↑]	2G1	[H-SRE-{1G1, 1G2, 1G3}↓] See: H-culture-{4G1, 4G2, 4G3, 6G3}; Appendix C.3.2-C.3.3; Appendix F, esp. F.1.
		2G2	Criteria for input validation are correctly established. See Appendix F.2 and F.4.
3	Incompleteness	3G1	See Table 1
		3G2	H-ProcState-3G5
		3G3	HA includes interactions with the environment of the system – see Section 3.4.1.
		3G4	Inter-relationships and interactions with the environment are analyzed in all configurations and modes (including degraded ones), and changes from one mode to another. [H-SR-3G3↑]
		3G5	In HA at system concept phase (Section 3.4), an architectural model or representation of the system (e.g., functional; behavioral) concept includes a (functional; behavioral) model or representation of the environment, especially the physical processes (Appendix H) [26]). [H-SR-3G3↑] [H-SAE-{1G1, 2G1, 3G1, 4G1}↓]
		3G6	Process behavior models[67] (H-SR-3G5) include identification of safe state regions and trajectory[68] of safely recoverable process state. [26], See Figure 5
		3G7	Process behavior models (H-SR-3G5) include time-dependencies, relationships and constraints. [11] [H5-G0]
		3G8	[H-SRE-{1G1, 1G2, 1G3}↓]
4	Inadequate protection or defense against residual	4G1	Monitoring: Feasible trajectories[70] of appropriate state variables[71] or parameters and expected values are known and monitored. (Generalized from [21] CP 2.1.3.2.3, 2.1.3.2.4)

[67] The scope is limited to I&C-relevance.

[68] State space within which recovery is provable.

[70] For example: Values over time; rate of change.

	faults[69]. [H-SR-3↑] Note tension with [H-SR-20]	4G2	Detection: Appropriate parameters of the system or element are monitored to detect departure from safe state (e.g., by applying discriminating[72] logic on the monitored parameters) in conjunction with predictive behavior models, but considering [H-SR-{19, 20}]
		4G3	Intervention: Upon detection of departure from safe state, intervention maintains the plant in safe state. (Adapted from [21] CP 2.2.3.7)
		4G4	Containment: The system or element is able to contain, localize, and isolate the source of the fault (e.g., a hardware or software component).
		4G5	Notification: Notification is timely, but avoids "flooding."
		4G6	[H-SRE-{1G1, 1G2, 1G3}↓]
5	Inadequate identification of sources of uncertainty, their effects, and their mitigation. [H-SR-3↑]	5G1	[H-SRE-{1G1, 1G2, 1G3}↓]
6	Deficiency in requirements for fault containment. [H-SR-3↑]	6G1	[H-SRE-{1G1, 1G2, 1G3}↓]
7	Inadequate or improper generalization to capture classes of issues	7G1	[H-SRE-{1G1, 1G2, 1G3}↓] See H-culture-{4G1, 4G2, 4G3, 4G4, 6G3} and Appendix F, esp. F.1.
8	Inconsistency	8G1	[H-SRE-{1G1, 1G2, 1G3}↓]
9	Inadequate protection or defense against invalid input [H-SR-4↑]		H_SR_2G2
		9G1	Validity of value of each input is monitored (in [21] CP 2.1.3.2.4).
		9G2	Intervention upon detecting invalid input is specified to maintain system safe state.
10	Uncorrected or inadequately compensated instrumentation errors	10G1	Required calibrations and corrections are known and applied (in [21] CP 2.1.3.2.5) [H-SR-9G1↑]
11	Implicit assumptions about the environment. [H-culture-12↑]	11G1	Each assumption about the environment is made explicit (e.g., documented; in [19] Appendix A.3 item 3). See Appendix J. [H-culture-{12G1-12G3}↑]
12	Invalid assumption about the environment.	12G1	See: Appendix J; H-culture-12G3; Appendix C.3.3; Appendix F.2.
		12G2	Each assumption about the environment is validated (e.g., through treatment as a "constraint or condition to be validated)."
13	Unclear expression of the consequences of an assumption [Table 1][H-SR-12↑]	13G1	Record of each assumption [H-SR-12G1] includes the consequences if the assumption turns out to be false. (In [19] Appendix A.3 item 4). Also see Appendix J.
		13G2	Requirements include measures to mitigate the consequences of assumptions that fail to hold. (In [19] Appendix A.3 item 4).

[71] Include inputs and outputs.

[69] It refers to faults internal to digital safety system and its elements; also known as resilience.

[72] e.g.: through infeasible or unexpected value.

		13G3	Each assumption (e.g., constraint or condition to be validated) is tracked as a configuration item.
		13G4	Assumptions about the downstream design are made explicit (e.g., through explicit derived requirements or constraints on the architecture, design and implementation, and the associated methods and tools). (In [19] Appendix A.3 item 3.1). See: Appendix J; H-culture-{12G2, 12G3}. Examples: 1. Requirements from the application software on system platform services (HW & SW), including HW and SW resources to support the workload. 2. Timing constraints to be satisfied. 3. Compatibility across maintenance updates.
		13G4.1	The safety plan and supporting plans include activities and tasks specifying how and when these assumptions will be validated.
14	Unmitigated consequence of invalid assumption	14G1	Record of each assumption [H-SR-12G1] includes how and when it will be validated. (In [19] Appendix A.3 item 3)
15	Incorrect order of execution or timing behavior [H-ProcState-1.3]	15G1	An explicit, verifiable (as determined through mathematical analysis) specification for the order of execution and timing inter-relationships, especially considering multiple concurrent physical processes, inter-process synchronization and shared resources (in [21] CP 2.1.3.2.2, 2.2.3.5). See Appendix I.
16	Inter-relationships and inter-dependence across requirements are not clearly understood or recognized [H0-4 – H0-8], resulting in unanalyzed conditions	16G1	Applicable types of dependencies across requirements are identified (see examples herein), modeled, and tracked. For example, if A and B are two requirements, their relationship types (See note) may be: • A requires B • B supports A • B hinders A • B is a selection for A (an exclusive one among many choices) • B is a specialization of A
		16G2	Hidden dependencies between functions (e.g., "unwanted feature interactions") do not exist.
17	Interference from unintended interactions or side effects. [H-S-1↑]	17G1	Interactions are limited provably[73] to those required for the safety functions.
		17G2	Absence of other unspecified behavior or side effects is assured.
18	Effects of sudden hardware[74] failure, esp. semiconductors	18G1	Requirements include failure or fault detection and containment measures, including offline ability to locate and isolate the source of the fault (e.g., a hardware or software component). [H-SRE-7G1↓]
Note for H-SR-16G1: Relationships may be one-to-one, one-to-many, many-to-one, and many-to-many.			
19	Allocated set of requirements leads to conditions that are unanalyzable or difficult to analyze.	19G1	[H-SRE-{1G1, 1G2, 1G3}↓]
20	Adding backups (or fault protection) can introduce new hidden dependencies and impair analyzability. [H-SR-19↑]	20G1	[H-SRE-{1G1, 1G2, 1G3}↓]

[73] Example: Unspecified interactions are not allowed.

[74] Also see Table 16

21	Although layered protection has benefits, there can be dilemmas from keeping software protected with several layers – analyzability may be impaired. [H-SR-19↑]	21G1	[H-SRE-{1G1, 1G2, 1G3}↓]
22	Inability to integrate correctly elements of a system (e.g., subsystems, hardware components, software components). [H-SR-{1, 2, 3, 8, 12, 13, 15, 16, 19↓] [H-SwR-2↓] [H-SRE-7↓] [H-SwRE-1↓] [H-HwP-1↓]	22G1	[H-SRE-{1G1, 1G2, 1G3}↓] [Table 14↓]
		22G2	There are no deficiencies in the specifications.
		22G3	There are no deficiencies in the elements to be integrated.
		22G4	The system is modularized properly, so as to allow concluding correctness from the correctness of the architecture and the correctness of the elements. H-S-1.1G1.4↑
23	Anomaly in the state of the process[75] is not recognized or identified or correctly understood or correctly specified. [H-SR-3↑] [H-SR-4↑]	23G1	See H-SR-3G6 The trajectory of safely recoverable process state variables (i.e., state space within which recovery is provable) is specified correctly. In other words, when departure from this state space or region is recognized, intervention can prevent departure from safe state. See Figure 5. See Appendix F.

3.5.1.3 Contributory hazards from system requirements engineering

The requirements engineering phase of the lifecycle is most sensitive to the quality of processes, including the resources applied. Requirements elicitation and analysis aspects are most sensitive to the competence [H-SRE-1] applied.

Table 10 identifies hazards contributed through weaknesses in the process of engineering requirements for the system.

Table 10: Inadequate system requirements engineering: Examples of contribution to hazards

Contributory hazard		**Conditions that reduce the hazard space**	
ID H-SRE-	**Description (e.g., Scenario)**	**ID H-SRE-**	**Description**

[75] The process that the safety system is observing or monitoring for safety-related intervention.

1	Inadequate competence [H-culture-6↑]	1G1	The team engaged in these activities is an assemblage of high competence in multiple disciplines, capable of creatively eliciting and synthesizing information from diverse sources, including implicit, experiential knowledge about the environment. The combined competence of the team matches the expertise needed in each phase in the engineering lifecycles, noting that the nature of expertise is not the same in all phases. See H-culture-{4G2, 4G3} and Appendix F.
		1G2	A different and independent diverse team reviews the requirements and their validation.
		1G3	The review team has expertise in discovering the types of mistakes or shortcomings identified in Table 9 and Table 10 H-SRE-{2-6}. See H-culture-6G3 and Appendix F. I item 5
2	Ambiguity in the natural language textual description [H-SAE-2↓]	2G1	A subset of the natural language is used such that requirements can be described unambiguously to the community of its users[76]; for example: 1. Closed set of language elements 2. Unambiguous semantics of each language element 3. Unambiguous compositions of language elements and their compositions Also see H-culture-{12G2, 12G3}. [H-SAE-1G1↓; H-SAE-1G2↓]
		2G1.1	Formal properties are abstracted, for later use in verification of next phase work product. [22][23]
		2G2	The language subset (H-SRE-2G1) supports distinct identification and description of the following: 1. Assumptions about the environment [26]; Appendix J. 2. Input from the environment (e.g., command, i.e., some signal requiring state-changing effect + required behavior), query, process state, other data. 3. Output (e.g., some signal having state-changing effect, state-notification, exception-notification) 4. Functions assigned to a human. 5. Procedure for the execution of each function assigned to a human (required behavior). 6. Other elements of the system being analyzed 7. Functions assigned to each element; required behavior. 8. Interactions required across elements 9. Constraints on the behavior and interactions of each element, e.g. timing constraints [11], Appendix I; QoS constraints. 10. Criteria to monitor and detect violation of a constraint [25].
3	Incorrect formalization from	3G1	[H-SRE-2G1↓; H-SRE-2G2↓] [H-SAE-1G1↓; H-SAE-1G2↓]

[76] Users include people and tools, employed in creation, modification, interpretation, transformation. maintenance, V&V, and regulation (adapted from CP 2.3.3.1.1 last sentence in [21]).

	intent or natural language text	3G2	Persons performing the task (see H-SRE-2G1.1) know the vocabulary of the application domain and know how to translate it into formal properties.
		3G3	1. Multiple independent persons/teams perform the task. 2. The discrepancies across their results are analyzed. 3. Another independent panel is engaged in resolving the discrepancies.
4	Input constraints are ambiguous.	4G1	Valid value type and range of each input are explicitly identified (in [21]CP 2.1.3.2.4). Also see Table 1. Also see H-culture-{12G2, 12G3}.
5	Loss of information in transfer and traceability of HA-results to requirements.	5G1	Activities of HA and Requirements Engineering are formally integrated (also see Table 1).
6	An atomic requirement is not traceable individually.	6G1	Each atomic requirement is traceable (in [21] CP 2.1.3.1; in [19] Appendix A.4 item 2) [H-S-2G4↓].
		6G2	Each requirement is a configuration controlled item[77].
7	Loss of information[78] across disciplines, processes, and organizational units (e.g., system engineering, software engineering, hardware engineering, safety, quality). [H-culture-9↑][H-SR-3↑] [H-SwRE-1↓]	7G1	Systems are engineered holistically, including crosscutting analysis. (Adapted from [19] Appendix A.3 footnote 82). See H-culture-9G1 and H-culture-9G2 and Appendix F. 1.
		7G1.1	The interaction across a system or an element and its environment is identified explicitly. Example: Models at every level of integration, such that the models are compatible and information can be integrated and analyzed across the various models.
			H_culture_{12G1, 12G2, 12G3}.

3.5.2 Software Requirements

Contributions to hazards through inadequacies in requirements at the system level (and corresponding conditions to reduce that hazard space) also apply to requirements for software. Even though correct, complete, consistent unambiguous requirements for software are supposed to flow down from the system engineering lifecycle, typically in practice, V&V for

[77] Other relevant references: IEEE 828 and 1042

[78] Current practice divides systems engineers, software engineers, and hardware engineers; often failures occur due to gaps in between. (From [19] Appendix A.3 footnote 82)

these properties occurs from the software engineering perspective[79] as a part of the software engineering lifecycle.

Some of the requirements from the system engineering lifecycle may be allocated directly (as is) to software. For other requirements from the system engineering lifecycle (e.g., quality requirements) additional requirements and constraints for software may be derived as part of the software engineering lifecycle. Also see Section 3.6 for constraints on software architecture. Contributory hazards and constraints identified in Section 3.6.1 for the system architecture also apply to software. Derived constraints on software design and implementation (D&I) are included in Sections 3.8 and 3.9.

[79] Focus: Check correctness of understanding; make explicit and unambiguous.

3.5.2.1 Contributory hazards in software requirements

The contributory hazards identified in Table 9 also apply to software requirements. Table 11 provides examples of additional hazards contributed through inadequacies in software requirements.

Table 11: Inadequacy in software requirements: Examples of contribution to hazards

Contributory hazard		**Conditions that reduce the hazard space**	
ID H-SwR-	**Description (**e.g., Scenario**)**	**ID H-SwR-**	**Description**
1	Inadequate flow-down of properties (Table 8) and other constraints from the system engineering lifecycle (Table 9) [H-SwR-2↓]	1G1	Corresponding constraints (Table 8; Table 9) are derived and applied to software
2	Inadequate flow-down of requirements & constraints to support integration of elements into a correctly working system.	2G1	Corresponding constraints (Table 8; Table 9) are derived and applied to software
3	Inadequate flow-down of requirements & constraints from NPP level to the safety system and then to its elements, including software.	3G1	Decomposed and derived requirements and constraints assure that their composition will satisfy the upstream (source) requirements (from which these were decomposed or derived) and not introduce unspecified behavior.
4	Software produces an output of infeasible value.	4G1	Appropriate conditions infeasible in the real world are identified and used to establish criteria to monitor for anomalous[80] behavior of software. (Adapted from [21] CP 2.3.3.1.5) , but not introducing adverse conditions as identified in H-SR-{19, 20}.

[80] Intent: Defend against weakness in requirements.

3.5.2.2 Contributory hazards from software requirements engineering

The contributions to hazards identified in Table 10 (and conditions to reduce the associated hazard space) also apply to software requirements engineering. Table 12 provides examples of additional contribution to hazards through inadequacies in engineering of software requirements.

Table 12: Inadequate software requirements engineering - contribution to hazards: Examples

Contributory hazard		**Conditions that reduce the hazard space**	
ID H-SwRE-	**Description (**e.g., Scenario**)**	**ID H-SwRE-**	**Description**
1	Loss of information across disciplines, processes, and organizational units (e.g., system engineering, software engineering, hardware engineering, safety, quality) due to discipline-wise division of organizations, people, and work [H-culture-9↑]	1G	H-SRE-{7G1; 7G1.1}↑ Also see H-culture-{9G1, 9G2, 12G2, 12G3} and Appendix F.
2	Loss of information across disciplines due to incompatible paradigms, methods, and tools across disciplines. [Example contributor: H-HwP-5↓]	2G	Methods and languages to describe or specify requirements allocated to software support unambiguous mapping and integration across dissimilar elements (e.g., interactions across hardware, software and human elements). [H-SAE-{2G1, 3G1}↑] [H-HwP-5G1↓] See Appendix F.4.

3.6 Evaluation of hazard analysis - Architecture

System failures traceable to architecture rank high in the experiences of various safety-critical, mission-critical, high-quality digital systems across a diverse range of application domains. For example, unwanted and unnecessary interactions, hidden couplings, feedback paths, and side effects have led to unexpected failures; verification based on traditional testing or simulation did not detect such flaws [19].

3.6.1 Contributory hazards in System Architecture

While the overall scope of system architecture includes the safety system under evaluation and its relationship with its environment, this section focuses on system-internal elements (e.g., hardware and software) and their inter-relationships (i.e., interfaces, interconnections, and interactions) whether these are direct or indirect, intended or unintended, explicit or implicit, static or dynamic, “normal” or “abnormal.”

The scope of system architecture activities includes the allocation of requirements and constraints to elements identified in the system architecture.

> Note: Architecture-specific evaluation of HA is predicated on the correct and complete performance of the overall HA, as illustrated in Table 1, including considerations of combinations of multiple contributory hazards, exemplified in

Table 2 through Table 7.

Table 8 and Table 13 include examples of contributors to hazards through system architecture and corresponding conditions that reduce the respective hazard spaces. These considerations are applicable to architecture-related contributory hazards in every phase in the development lifecycle (from conception to implementation), to every level in the integration hierarchy, and to transformations from one level to another. Thus, the information in these tables should be applied to the context of the level of integration being analyzed.

Table 13: Interference: Example scenarios and conditions that reduce the hazard space

Contributory hazard		**Conditions that reduce the hazard space**	
ID H-SA-	**Description (**e.g., Scenario**)**	**ID H-SA-**	**Description**
3	Scenario: A system, device, or other element (external or internal to a safety system) may affect a safety function adversely through unintended interactions, caused by some combination of defects, deficiencies, disorders, malfunctions, or oversights. [H-SR-17↑]	3G1	[H-SR-17G1↑]
		3G2	Interactions and interconnections that preclude complete[81] V&V are avoided, eliminated, or prevented. (CP 2.2.3.11 in [21])
		3G3	Freedom from interference is assured provably[82] across: 1. Lines of defense [34] 2. Redundant divisions of system (CP 2.2.3.6 in [21] 3. Degrees of safety qualification[83] (CP 2.2.3.3 in [21]) 4. Monitoring & monitored elements of system.
		3G4	Analysis of the system demonstrates that unintended behavior is not possible[84]. 1. Interaction across different sources of uncertainty is avoided. 2. The architecture precludes unwanted interactions and unwanted, unknown hidden coupling or dependencies (in [19] Appendix A.4 item 6). 3. Specified information exchanges or communications occur in safe ways (in [19] Appendix A.4 item 6).
		3G5	Only well-behaved interactions are allowed [H-S-1.2G{1,2,3}, H-S-1.3G{1,2}↑]
		3G6	Constraints are identified for such contributing hazards from the environment as electromagnetic interference – see examples in Appendix E.4.
		3G7	The impact of dependency-affecting change is analyzed to demonstrate no adverse effect. [Table 1]
4	Scenario [H-S-3G4↑]: A function, whose execution is	4G1	Analysis of the execution-behavior of the system proves that such interference will not occur. For example, worst-case

[81] "Completeness" includes confirmation that all specified requirements have been satisfied and confirmation that the requirements are correct, complete, consistent, and unambiguous.

[82] Example: There is no pathway.

[83] In other application domains, the corresponding concept is known as "mixed criticality."

[84] Example: There is no pathway.

	required at a particular time, cannot perform as required, due to interference through sharing of some resource it needs.		execution time is guaranteed.
5	Timing constraints are not correctly specified and not correctly allocated.	5G1	Timing requirements for monitoring a continuously-varying phenomenon are derived, specified, and allocated correctly to the services and elements upon which their satisfaction depends. Example: Sampling interval that characterizes the monitored variable with fidelity.
		5G1.1	Commensurate required sampling interval is determined through mathematical analysis.
		5G1.2	Discretization and digitization do not affect the fidelity required, as determined through mathematical analysis.
		5G1.3	Aliasing is avoided.
		5G1.	Sampling periods to monitor discrete events are established correctly, as determined through mathematical analysis.
6	Sampling and update intervals are not commensurate to the timing constraints of the associated control actions. [H-SR-15]	6G1	Update intervals support the timing constraints of the required control actions, as determined through mathematical analysis.

3.6.2 Contributory hazards from system architectural engineering

Applying the reference model depicted in Figure 4 to the activities of architectural engineering, Table 14 identifies hazards contributed through some of the resources and elements employed in these activities and commensurate constraints on these process activities. Additionally, as stated in Section 3.7, considerations therein "are applicable to architecture-related contributions to hazards in every phase in the development lifecycle (from conception to implementation), to every level in the (system, subsystem, component, sub-component …) integration hierarchy, and to transformations from one level to another."

Table 14: Inadequate system architectural engineering: Examples of contribution to hazards

Contributory hazard		**Conditions that reduce the hazard space**	
ID H-SAE-	**Description (**e.g., Scenario**)**	**ID H-SAE-**	**Description**
1	The architecture[85] description (including requirements allocated to its elements) is ambiguous, rendering it vulnerable to interpretations other than intended. For example, textual descriptions use words and expressions and graphic representations use symbols, for which commonly understood meanings have not been agreed upon by the community of its users. [H-S-2G-6↑] [H-SAE-2↓; H-SAE-3↓]	1G1	Description method supports distinct, unambiguous description of the following: 1. Assumptions about the environment. 2. Input from the environment (e.g.: command (some signal requiring state-changing effect + required behavior); query; data. 3. Output (e.g., some signal having state-changing effect), state-notification, including exception-notification. 4. Functions assigned to a human 4.1. Procedure for the execution of each function assigned to a human (required behavior) 5. Other elements of the system 5.1. Functions assigned to each element; required behavior. 6. Inter-relationships of elements. 7. Interactions required across elements. 8. Constraints on the behavior and interactions of each element, e.g. timing constraints – Appendix I; QoS constraints. 9. Criteria to monitor and detect violation of a constraint.
		1G2	The language (graphic or text-based) used in the description or specification is unambiguous; for example: 1. Closed set of language elements. 2. Unambiguous semantics of each language element. 3. Unambiguous semantics of the compositions (e.g., rules of composition) of language elements and their compositions.
		1G3	The method and language are applied correctly.

[85] The term is used in its comprehensive sense, e.g., it includes conceptual architecture (or requirements architecture), system design architecture, software design architecture, hardware design architecture, software implementation architecture, function/procedure-architecture.

2	Transformation, refinement or elaboration of architecture from one lifecycle phase to another does not preserve semantics and leads to unintended behavior.	2G1	Methods and languages to describe, represent, or specify architectures (including requirements allocated to various elements) support unambiguous transformations or mappings across architectural artifacts (e.g., transformation from system conceptual or requirements level to system design level to software design level to software implementation level to procedure or subroutine or function level).
		2G2	Information is used with semantic consistency across different elements of the system.
3	When dissimilar elements are integrated (have to work together), their interaction leads to unintended behavior, due to semantic mismatch (e.g., a signal from a sender does not have the same meaning for the receiver).	3G1	Methods and languages to describe, represent, or specify architectures (including requirements allocated to various elements) support unambiguous mapping and integration (including composability and compositionality for essential properties) across dissimilar elements (e.g., interactions across hardware and software elements).
		3G2	Information is used with semantic consistency across different elements of the system.
4	When elements from different sources or suppliers are integrated (have to work together), their interaction leads to unintended behavior, due to semantic mismatch (e.g., a signal from a sender does not have the same meaning for the receiver).	4G1	Methods and languages to describe, represent, or specify architectures support unambiguous transformations or mappings and integration (including composability and compositionality for essential properties) across elements from different sources or suppliers.
5	A tool used in architectural engineering is not qualified to produce, manipulate or handle a safety grade architectural artifact (e.g., system, element, and data).	5G1	Each tool is qualified for safety grade use.
		5G2	Restrictions necessary for safe use of a tool are identified and the set of restrictions, tracked as a configuration controlled item.
6	Tools used in engineering a system, engineering software, or engineering hardware do not integrate correctly, that is, semantics may not be preserved in information exchanged across the tools.	6G1	Tools intended to be used collectively or in an integrated process are configured and qualified for safety grade use as a set, tracked as a configuration controlled item.
		6G2	Restrictions on individual tools, their information exchange functions, and their interactions, which are needed for safe use of the tools as a set, are identified and the set of restrictions, tracked as a configuration controlled item.
		6G3	Semantics of the information accepted and provided by the tools are explicitly represented.
7	A reused element (e.g., from some previous project or system; previously verified to satisfy its specifications), when integrated in this system, does not provide the intended system behavior (e.g., semantics may not be preserved in the flowdown of specifications or their realization).	7G1	Pre-existing element is qualified for the environment[86] in which it is to be reused.
		7G1.1	Allocation of requirement specifications from system to the element is validated to be correct.
		7G1.2	Pre-existing specification of the element satisfies the requirement specification allocated from this system.
		7G1.3	The element satisfies the allocated requirements specification

[86] including assumptions about the environment – also see H-culture-12

		7G2	Restrictions on the use of a pre-existing element in the target environment are identified and the set of restrictions, tracked as a configuration controlled item.
7.1	Some assumption about the reused element or its usage environment is violated. Also see H-SR-13. [H-culture-12]	7.1G1	H-ProcState-4G1.2; H-culture-12G1; H-SR-13G3
8	Individuals performing architectural engineering functions may not be cognizant of the usage-limitations of the tools, elements, and artifacts accessible to them.	8G1	Human resources employed in architectural engineering are qualified to perform their roles, especially usage-limitations of the tools, elements, and artifacts available to them, commensurate to the overall complexity of the cognitive activities to be performed.

3.6.3 Contributory hazards in Software Architecture

The information in Section 3.6.1 and Table 8 and Table 13 also applies[87] to software architecture, esp. relationships of software with its environment (e.g., hardware elements and human elements). This section focuses on software elements that are internal to the safety system and their inter-relationships, i.e., interfaces, interconnections, and interactions, whether these are direct or indirect, intended or unintended, explicit or implicit, static or dynamic, "normal" or "abnormal[88]."

The scope of software architecture activities includes the allocation of requirements and constraints to elements identified in the software architecture.

> Note: The contents of this section are predicated on correct performance of HA, as discussed in preceding sections and complete satisfaction of the criteria to prevent, avoid, eliminate, contain, or mitigate the categories of hazards identified in those sections.

These considerations are applicable to architecture-related contributory hazards in every phase in the software development lifecycle (from conception to implementation), to every level in the software integration hierarchy[89], and to transformations from one development phase or level to another.

Table 15: Contribution to hazards through software architecture: Examples

Contributory hazard		**Conditions that reduce the hazard space**	
ID H-SwA-	**Description (**e.g., Scenario**)**	**ID H-SwA-**	**Description**
1	Scenario: Software contributes to or exacerbates complexity of the system, making it difficult to verify [H-S-1.1↑] and understand [H-S-2↑].	1G1	The behavior of a non-atomic element is a composition of the behaviors of its constituent elements, with well-defined unambiguous rules of composition[90]. (In [19] Appendix A.4 item 5) 1. Interfaces of elements are unambiguously specified, including behavior (adapted from [19] CP 2.3.3.2.2 last sentence).

[87] Replace "system" with "software" or consider the scope of the system to be narrowed down to software.

[88] Examples: Invalid input; hardware malfunction; human mistake.

[89] Examples: Subsystem; module; subroutine.

[90] Including conditions for composability and compositionality for required properties.

			2. Interactions across elements occur only through their specified interfaces, that is, adhering to principles of encapsulation (adapted from [21] CP 2.3.3.2.2).
		1G2	The system is modularized using principles of information hiding and separation of concerns, avoiding unnecessary interdependence (in [19] Appendix A.4 item 7).
		1G2.1	Corresponding specifications are modularized.
		1G2.2	Corresponding specifications, plans, and procedures for verification are modularized.
		1G3	Each element (e.g., a software unit) is internally well-architected (that is, satisfying conditions stated earlier), such that its properties [Table 4] can be assured. For example: 1. A software unit implementing some NPP safety function(s) is composed from semantically unambiguous atomic functions and data using well-defined unambiguous rules of composition. [H-SwA-1G1↑] 2. Paths from inputs to outputs avoid unnecessary coupling. [H-SAE-1G2↑] 3. Unnecessary remembering of state information across execution cycles is avoided. (Adapted from CP 2.3.3.2.8 in [19])
2	Order of execution or timing behavior are not analyzable correctly, because of system complexity	2G1	Complexity-increasing behaviors are avoided [H-S-1.1.1G1↑]; simplicity-increasing features are preferred; for example: 1. Static configuration of tasks[91] to be executed (adapted from [21] CP 2.4.3.8.1 2nd and 3rd bullets). 2. Tasks in execution are run to completion[92] (adapted from [21] CP 2.4.3.8.1 1st bullet). 3. Static allocation of resources[93] [H-S-4G1↑] (Generalized from [21] CP 2.4.3.8.1 4th bullet).
3	Behavior is not analyzable mathematically or analysis is not mechanize-able for lack of a semantically adequate paradigm or model underlying the behavior specification or description. [H-SAE-{1,2,3}]	3G1	Behavior specification or description method is based on a semantically adequate, unambiguous paradigm [H-SAE-1G1↑; H-SAE-1G2↑], supporting association of timing constraints [H-SR-13G4↑], other properties (Table 8↑), hierarchical nesting, and abstraction [H-S-1.1G1↑]. Example paradigm: Extended finite state machine (adapted from [26] and 2.3.4.1.1 in [21]).

3.6.4 Contributory hazards in Software architectural engineering

Table 14 is also applicable to the architectural engineering of software, with software-related refinements added in Table 16.

Table 16: Hazards through inadequacy in software architectural engineering: Examples

Contributory hazard	Conditions that reduce the hazard space

[91] Task: Schedulable unit of work (execution). Dynamic creation and destruction of tasks is avoided

[92] Example: Interruption and pre-emption are avoided or mathematical analysis (Appendix I) proves satisfaction of constraints on timing and order of execution.

[93] Examples: Memory (information storage); Processor (execution time)

ID H-SwAE-	Description (e.g., Scenario)	ID H-SwAE-	Description
1	Loss of information across disciplines (e.g., system engineering, software engineering, and hardware engineering) due to discipline-wise division of organizations, people, and work [H-culture-9↑].	1G	H-SRE-tG1↑
2	Loss of information across disciplines due to incompatible paradigms, methods, and tools across disciplines.	2G	H-SAE-{2G1, 3G1}↑

3.7 Evaluation of Hardware-Related Hazard Analysis

As in the preceding sections, hardware-related HA is treated in two parts – the product (Table 17: Hardware: Examples of contribution to hazards) and the process (Table 18: Inadequate hardware engineering: Examples of contributory hazards).

Table 17: Hardware: Examples of contribution to hazards

Contributory hazard		Conditions that reduce the hazard space	
ID H-Hw-	Description (e.g., Scenario)	ID H-Hw-	Description
1	Failure of hardware leads to unanalyzed conditions [H-S-1.1.1↑] (e.g., unknown state).	1G1	Only hardware with predictable, well-understood, well-known degradation behavior is used.
		1G2	Degradation is detectable before failure that could lead to unanalyzed conditions (e.g., unknown state) [H-S-1.2G3↑]. (Adapted from CP 2.2.3.7 1st clause in [21])
		1G3	Safety requirements are specified to maintain system in a safe, known state at all times, in all modes of usage, including degraded states and including maintenance. Safety functions may be online or offline; for example: 1. Monitor hardware condition [H-SR-4G1↑]; for example: 1.1. Online monitoring (e.g., cyclic; periodic) 1.2. Offline surveillance 2. Detect hardware fault [H-SR-4G2↑] – see H-Hw-1G4 3. Notify (other automation or human) [H-SR-4G5↑] 4. Intervene (to maintain system in safe state) [H-SR-4G3↑] 5. Perform preventative maintenance (e.g., scheduled replacement) 6. Provide redundancy 6.1. Provision of diverse redundancy (Items 1-4 adapted from CP 2.2.3.7 in [21]); (Item 4 is generalized from CP 2.2.3.7 in [21])

		1G4	Requirements are identified for independent, timely detection of a contributory hazard in an instrument or other element upon which a safety function depends; for example: 1. In the case of a bi-stable device, the device can be feasibly in one stable state or the other only; then, an indication of both states at the same time is an anomaly. 2. In the case of a continuously controlled electric motor for a motor-operated valve, if the trajectory {electric current; displacement; time} for the transition from actuation command to completion is outside the envelope of feasibility, it indicates an anomaly. 3. The trajectory of feasible process state variables (set of values over time) is identified, such that indication of an instrument anomaly can be derived from sensed values in the infeasible region.
2	Anomaly in the state of the process is not recognized or identified or correctly understood due to inadequacy in instrumentation [H-SR-23↑]	2G1	Progressive degradation, drift, and such other changes in the behavior of instrumentation are properly accounted for; examples: 1. Monitoring and tracking such phenomena 2. Compensation 3. Calibration; recalibration; 4. Allowances (margins) for unaccounted, uncompensated, or unknown changes 5. Detection of unacceptable deviation 6. Appropriate intervention – see H-Hw-1G3 items 2,4.
3	Anomaly in the state of the instrumentation for the safety functions or other element in the environment, upon which a safety function depends, is not correctly understood or recognized.	3G1	Instrument or element has behavior (including behavior in fault states), which satisfies requisite properties such as those identified in Table 8.
4	Loss or interruption of power.	4G1	Safety functions are specified to maintain system in a safe, known state (adapted from CP 2.2.3.7 last sentence in [21]).
5	Disturbance in power supply.		
6	Inadvertent alteration of invariant information (e.g., program code; fixed data).	6G1	Invariant information is stored in read only memory (ROM). (Adapted from CP 2.7.3.3.2 in [21]).
7	Change in hardware that is nominally “equivalent” to replaced hardware (e.g., functionally, electrically, mechanically “interchangeable”) leads to some subtle change that degrades a safety function.	7G1	Criteria for equivalence are correct and complete; examples: 1. Analysis of differences in timing behavior. 2. Analysis of differences in signal-noise discrimination. Also see Table 1. 3. If the item includes programmable logic, analysis for its contribution to hazards.

Table 18: Inadequate hardware engineering: Examples of contributory hazards

Contributory hazard		Examples of conditions that reduce the hazard space	
ID HwP-	**Description (**e.g., Scenario**)**	**ID H-HwP-**	**Description**
1	Loss of information across disciplines (e.g., system engineering, software engineering, and hardware engineering) due to discipline-wise division of organizations, people, and work [H-culture-9↑].	1G1	H-SRE-tG1↑ See H-culture-{4G1, 4G2, 4G3, 4G4} and Appendix F.
2	Loss of information across disciplines due to incompatible paradigms, methods, and tools across disciplines.	2G1	H-SAE-{2G1, 3G1}↑
3	Preventative maintenance activities on which a safety function is dependent are not performed [27] when needed or scheduled [H-Hw-1G3].	3 G1	Maintenance schedules specify the preventative actions explicitly [H-Hw-1G3↑].
		3G2	These maintenance schedules are treated as safety related activities (e.g., including, performance; verification; audit) [Table 1].
4	Preventative protection against age-related degradation is not provided in maintenance plans (generalization from [28]).	4G	[see H-Hw-{1G1; 1G2}]
5	Computation is incorrect due to incorrect mapping of algorithm onto arithmetic hardware; for example, due to incompatibility in one or more of the following: 1. Hardware 2. Hardware interfacing software 3. Algorithm 4. Mapping algorithm software onto hardware 5. Associated library software [H-SwRE-2↑]	5G1	The hardware (e.g., floating point processor), algorithm (e.g., formula and data types in the application software), and the transformation (e.g., compiler and its configuration) are specified correctly. (Generalized from CP 2.4.3.5.8 in [21]).
		5G2	The hardware, software, and transformation are qualified and configured correctly for conformance to the specs (H-HwP-5G1).
6	Selection of output destination (e.g., actuator) or input source (e.g., sensor) is incorrect, for example, due to incorrect mapping from software to hardware.	6G1	I/O-identifying mappings from requirements to architecture to detailed design to implementation are verified to be correct. (Generalized from CP 2.3.3.1.7 1st sentence in [21]).

3.8 Evaluation of Hazard Analysis related to Software Detailed Design

Review of HA under 10 CFR Part 52 is limited to review of work products from the pre-certification phases of the lifecycle (e.g., plan; concept; requirements; architecture). However, these work products could also include other constraints remaining after design certification for preventing contribution to hazards from activities in the later phases. Then, these constraints could be identified as part of the licensing basis, and could become part of ITAAC commitments.

Many defects found during software detailed design are traceable to (rooted in) deficiencies from earlier phases in the development lifecycle. Earlier sections of this RIL have identified examples of those deficiencies as contributory hazards. Those conditions to reduce the respective hazard spaces also apply to software detailed design.

Table 19: inadequate detailed design of software: Examples of contribution to hazards

Contributory hazard		**Examples of conditions that reduce the hazard space**	
ID H-SwD-	**Description (**e.g.,: Scenario**)**	**ID H-SwD-**	**Description**
1	Loss of information across disciplines (e.g., software architecture engineering and detailed software design). [H-SwAE-1↑]	G1	H-SAE-{2G1, 3G1}↑
2	Scenario: Software contributes to or exacerbates complexity of the system, making it difficult to verify [H-S-1.1↑] and understand [H-S-2↑]. [H-SwA-1↑]	G2	
3	Names of functions, data items, inputs, outputs, and variables in software are such that it becomes difficult to trace back to system requirements and further back to the application domain. (Adapted from [21] 2.3.4.1.2).	G3.1	Naming conventions and data dictionaries are established for ease of comprehension and bidirectional traceability.
		G3.2	Naming conventions and data dictionaries are used consistently.

3.9 Evaluation of Hazard Analysis related to Software Implementation

Many defects found during software implementation (coding) are traceable to (rooted in) deficiencies from earlier phases in the development lifecycle. Earlier sections of this RIL have identified examples of those deficiencies as contributory hazards. The conditions to reduce the respective hazard spaces affect software implementation also.

Common Vulnerabilities and Exposures (CVE) [31] and Common Weakness Enumeration (CWE) [32] are forms of contributory hazards in computer programs. Safe programming languages or safe subsets of appropriately selected programming languages reduce these hazard spaces effectively.

Table 20: Hazards contributed in software implementation: Examples

Contributory hazard		**Conditions that reduce the hazard space**	
ID H-SwI-	**Description (**e.g., Scenario**)**	**ID H-SwI-**	**Description**
1	Behavior is not analyzable mathematically or analysis is not mechanize-able, due to the complexity introduced through the improper use of interrupts or other mechanisms affecting order of execution.	1G1	Unnecessary use of interrupts is avoided, for example, not using interrupts to cover for inadequately understanding timing behavior of the physical phenomena (Table 1; H-SR-3G7) or the design and implementation (H-SR-13G4, H-SR-15G1)
		1G2	Schedulability analysis or proof is provided to verify that timing behavior of the implementation satisfies the specifications (H-SR-15G1).
2	Timing problems prevent deterministic behavior. Timing problems are difficult to diagnose and resolve.	2G1	The results produced by the programmed logic is not dependent on either: – the time taken to execute the program, or – the time (referenced to an independent "clock") at which execution of the program is initiated. (Adapted from [33])
		2G2	Execution speed does not affect correct order of execution.

4 Discussion of regulatory significance

Hazard analysis of a digital safety system[94] could address clause 4.8 (quoted below) in [3], where a "condition having the potential for functional degradation of safety system performance" is a hazard and a "provision … incorporated to retain the capability for performing the safety functions" is a requirement or constraint to eliminate, prevent or otherwise control the hazard.

> Clause 4 and sub-clause 4.8 in [3] A specific basis shall be established for the design of each safety system of the nuclear power generating station. The design basis shall also be available as needed to facilitate the determination of the adequacy of the safety system, including design changes. The design basis shall document as a minimum …:
>
> 4.8. The conditions having the potential for functional degradation of safety system performance and for which provisions shall be incorporated to retain the capability for performing the safety functions …

Hazard analysis of a digital safety system could support the "analysis…of the major structures, systems, and components…" required per 10 CFR 50.34(a)(3) as follows: HA could support the development of principal design criteria and derivation of design bases from these criteria [35] and corresponding clause 10 CFR 52.47(a)(2) of [36] "… analysis of the structures, systems, and components (SSCs) of the facility, with emphasis upon performance requirements, the bases, with technical justification therefor, upon which these requirements have been established, and the evaluations required to show that safety functions will be accomplished…. The description shall be sufficient to permit understanding of the system designs and their relationship to the safety evaluations." Hazard analysis of a digital safety system coul;d be part of the "analysis…of the major structures, systems, and components…" Hazard analysis of a digital safety system identifies design characteristics and unusual or novel design features, and associated principal safety considerations. In this way the hazard analysis of a digital safety system could support requirements of clause 5.6 in [3], which is dependent upon clause 4.8, by yielding principal design criteria, design bases, and derived requirements and constraints relating to independence with the specificity needed for consistent verification and validation.

Recognizing from recent licensing review experiences, trends in design characteristics and unusual or novel design features, generally accepted engineering standards[95] are not sufficiently specific to ensure consistent application and require significant judgment relying on high level of subject matter competence. In consideration of these trends and similar trends in other application domains and issues encountered in respective safety reviews, this RIL identifies the associated contributory hazards and corresponding system characteristics and conditions that reduce the respective hazard spaces. In turn, this could reduce the judgment space in regulatory evaluation and thus, regulatory uncertainty perceived by the applicant.

In support of requirements in 10 CFR 50.34(a)(3)(i) and 10 CFR 52.47(a)(3)(i), hazard analysis of a digital safety system could lead to principal design criteria, additional[96] to or overlapping the

---

[94] A system to which a safety function has been allocated as a result of a plant level safety analysis, which includes a plant level hazard analysis.

[95] It refers to their mention in 10 CFR 50.34(a)(ii)(B), and include standards referenced in NRC's regulatory guides

[96] These additional requirements or constraints may be specific to a facility, system, component or structure.

general design criteria (GDCs) in 10 CFR 50 Appendix A, which provide only minimum requirements.

In support of requirements in 10 CFR 50.34(a)(3)(ii) and 10 CFR 52.47(a)(3)(ii), hazard analysis of a digital safety system could lead from principal design criteria to design bases, that is, functions to be performed (functional requirements) and restraints (e.g., constraints on the architecture, and constraints on design and implementation), such that their satisfaction is verifiable later in the system development lifecycle. These derived requirements and constraints lead to the level of design information to which the following requirement in10 CFR 52.47 refers:

> "*The application must contain a level of design information sufficient to enable the Commission to judge the applicant's proposed means of assuring that construction conforms to the design and to reach a final conclusion on all safety questions associated with the design before the certification is granted. The information submitted for a design certification must include performance requirements and design information sufficiently detailed to permit the preparation of acceptance and inspection requirements by the NRC…"*

In support of requirements in 10 CFR 50.34(a)(4), hazard analysis of a digital safety system could be part of the preliminary analysis which yields principal design criteria, design bases, and derived requirements and constraints with the degree of specificity needed for consistent verification and validation. Hazard analysis naturally organizes this information along flow-down (or dependency) paths from a safety function, since it follows a cause-effect course of enquiry and reasoning, originating from potential for degradation of the safety function. This cause-effect course of enquiry and reasoning could also support developing specific information required per 10 CFR 50.34(a)(5-8) and 10 CFR 52.47(a)(7, 19), where critical to safety analysis.

The technical basis and safety goal-focused organizing framework established in RIL-1101 contributes limited support for risk-informed treatment as follows. It contributes to the determination of safety significance through systematic identification of a hazard, i.e., potential for adverse effect on a safety function allocated to the system under evaluation. This approach also supports identification of contributors to a hazard; for example, potential for adverse effect on diversity or defense-in-depth.

5 Conclusions

This RIL provides the US Nuclear Regulatory Commission (NRC)'s licensing staff the technical basis to support their review of hazard analysis (HA) performed on a digital safety system by an applicant seeking a design certification or a license amendment.

The RIL has been focused on certain kinds of issues encountered in NRO's recent licensing reviews, which are rooted in systemic causes, and are contributed through engineering deficiencies during the development of a digital safety system – characterized as contributory hazards; for example: Unintended or unwanted interactions; Inadequate definition of the boundary of the digital safety system being analyzed; incorrect decomposition and allocation of constraints to control hazards from the top-level of a digital safety system flowing down the integration hierarchy; inadequate flow-down to identify requirements and constraints on technical processes, supporting processes, and organizational processes.

Although the targeted scope was limited, the result supports a broader purpose. Hazard analysis organizes information along cause-effect dependency paths (Table 1; items H-0-8, H-0-9; Appendix K) from a safety function to a contributing item, and provides a framework for

reasoning about the (perceived) deficits (Appendix C3.3). In this manner, it contributes to risk-informed evaluation of the system.

The cause-effect dependency network resulting from hazard analysis provides a safety-goal focused organizing framework, which an applicant could use to streamline its safety analysis report, justifying elimination of those provisions in NRC-referenced standards which do not contribute to the safety goal. The applicant could also use this framework to justify alternative ways of satisfying NRC's regulation, where the applicant's approach is not aligned with the NRC's current guidance or standard review plan, but meets the safety goal. The applicant could also use this methodology to analyze modification to an existing I&C safety system (e.g., replacement of an older-technology module with a newer digital technology module), and use the resulting requirements and constraints to drive the modification.

Currently, different sets of regulatory guidance exist for power reactors, nonpower reactors, research and test reactors, and nuclear material processing facilities. The organizing framework introduced in this RIL opens opportunities to harmonize and streamline[97] the different sets of regulation, without increasing the burden of preparing an application or a safety analysis report for any particular type of system.

This organizing framework leads the way to an improved safety-focused future regulatory framework, as discussed in the next section.

This study found very little published information organized specifically to support HA reviews applicable to the targeted scope. Therefore, information assimilated in the RIL includes knowledge acquired through consultation with external experts. Through this process, RIL-1101 presents a unique assimilation of the state of the art. This technical basis supersedes that given in [39].

6 Future research, development and transition

The development of this RIL has opened many opportunities to improve the effectiveness and efficiency of the regulatory review process for digital safety systems, as identified below for future consideration in accordance with the priorities of the licensing offices and the availability of resources.

6.1 Transition, knowledge transfer and knowledge management

The trend towards systems with increasing interactions, fostered through networks and software, has rendered traditional hazard analysis techniques, such as FMEA and FTA, inadequate. Whereas other techniques (Appendix C.6), more suitable for this trend, have been known for some time, these are less familiar to the NPP industry, including NRC's licensing reviewers. Plans are underway to make this knowledge more easily deployable in practice, including illustrative examples. Consistent with recommendations in [40] about domain-specific software engineering, future R&D activities will investigate techniques to represent the knowledge of the domain in a form that is easy to find and reuse correctly.

NRC will coordinate its plans with EPRI, in order to share the knowledge base that is common across various stakeholders' activities: System development by the applicant or its supplier; safety analysis by the applicant; safety evaluation by the regulatory reviewer.

[97] For example, in the concept of "item relied on for safety (IROFS)" used in nuclear material processing equipment, the "relied on" relation maps into a dependency relation, explained in Appendix K of RIL-1101.

Transition plans will include learning cycles through pilot applications (a suggestion that came from the ACRC I&C Sub-committee). RES support for pilots will be defined in conjunction with the licensing offices.

6.2 Integration of safety significant information from NPP level analysis

The trend towards systems with increasing interactions, fostered through networks and software, increases the difficulties of analyzing dependencies of a safety system on conditions in its environment. For example, the traditional individual FMEA of other I&C SSCs does not suffice. With the commensurate growth in the volume of information, traditional manual methods will not be scalable. Information-sharing and consistency-maintenance will require mechanized support. Future R&D and transition plans will include investigation of more effective methods.

6.3 Harmonization and disambiguation of vocabulary

Differences in vocabulary hamper NRC's learning from NPP experiences elsewhere in the world and from other application sectors. The same terms have different meanings. The same concepts have different terms. Different concepts are mixed in different ways and enwrapped in different terms. These conditions lead to ambiguities and unnecessarily encumber the tasks of safety analysis and evaluation.

Future R&D will explore ways and means to bridge these communication gaps (e.g., modeling knowledge of the domain, as mentioned in Section 6.1). NRC will coordinate its plans with EPRI.

6.4 International harmonization

Different regions of the world pursue the same or similar safety goals under different regulatory and guidance frameworks, referencing different standards. These differences obstruct reaching a common understanding of the issues and establishing common or harmonized evaluation criteria. The NRC's current guidance is closely tied to legacy standards, which are not able to keep up with the changing technological environment. The safety-goal focused organizing framework introduced in this RIL opens an opportunity to remove this obstacle. Building on the vocabulary harmonization effort mentioned in Section 6.3, future R&D will explore international harmonization of the technical basis for evaluating hazard analysis.

6.5 Learning from other application domains and agencies

Other regulated application domains, such as life-critical medical devices and mission-critical flight control systems are experiencing the same trend towards systems with increasing interactions, fostered through networks and software. Larger markets than nuclear power are driving regulatory practices in those domains. In accordance with executive guidance, future R&D will include coordination with such other regulators and with other federal agencies, exploring the leveraging of a common R&D infrastructure [37], and approaches to address safety and security assurance earlier in the system development lifecycle [38].

6.6 Analysis earlier in the system development lifecycle

In the case of new reactors, applications for design certification have been based on process conformance rather than evidence about the design of the system such as architectural specifications and constraints on subsequent detailed design and implementation. Appropriate

architectural design and analysis requires abstractions that have not been a part of common practice in the NPP industry. However, architectural design and analysis is being used in other critical application domains. Future R&D and transition plans include making it easier to introduce that knowledge in the NPP application domain, building on the R&D mentioned in Sections 6.1 and 6.3.

6.7 Risk-informed evaluation

Future R&D will investigate hazard analysis methods applicable to risk-informed evaluation of systems in which safety significant conditions can arise from unintended interactions, engineering deficiencies, or other such systemic causes. For example, investigation will include methods to model and analyze dependencies.

6.8 Integrated hazard analysis for safety, security and other concerns

The organizing framework introduced in this RIL opens an opportunity to extend the design review for safety to include hazards from breach in security in the digital realm and to include hazards contributed through considerations of non-safety objectives driving a safety system configuration.

6.9 Integrated assurance framework

The organizing framework established through hazard analysis, as treated in this RIL, provides a logical framework to integrate the results of verification activities, as explained in Section 2.7.8 through Figure 1 and Appendix C.3 through Figure 10. This basis feeds into a related ongoing research activity to understand how a better "safety demonstration framework" (e.g., an assurance case framework) could address issues experienced in regulatory reviews in different regions of the world. Through the OECD/NEA Halden Reactor Project, NRC is collaborating with other regulatory experts to identify common needs and a common technical basis to meet these needs. The intent is to shift the paradigm from clause-by-clause compliance with regulatory guidance to meeting the safety goal. It is envisioned that the same framework could be applied to any level of integration within a digital safety system (e.g., embedded digital devices). It is expected that this framework would also provide efficient support for sustenance after a new reactor becomes operational, e.g. modification[98]. Review comments from external experts include recommendations to adopt an assurance case framework.

6.10 Ideas received through review comments

Suggestions and remaining issues identified in review comments are treated as inputs to NRC's next I&C research plan. For example, external expert review suggestions include

1. Additions for hazards contributed through tools.
2. Extension of the content concerning detailed design and implementation.
3. Additions for hazards contributes through FPGA and CPLD implementations.

[98] Intent of §10 CFR 50.59

7 Abbreviations and Acronyms

ACRS	Advisory Committee on Reactor Safeguards
CFR	Code of Federal Regulations
CP	common position[99]
CPLD	complex programmable logic devices
DI	design and implementation
DI&C	digital instrumentation and control
FPGA	field programmable gate array
FMEA	fault modes effects and analysis
FTA	fault tree analysis
GDC	general design criteria
HA	hazard analysis
HAZOP(S)	hazard and operability studies
I&C	instrumentation and control
IT	information technology
ITAAC	inspections, tests, analyses, and acceptance criteria
NPP	nuclear power plant
NRC	U.S. Nuclear Regulatory Commission
NRR	Office of Nuclear Reactor Regulation
NRO	Office of New Reactors
PHA	preliminary hazard analysis
QoS	quality-of-service
R&D	research and development
RAI	request for additional information
RIL	research information letter
SAR	safety analysis report
SER	safety evaluation report
SRP	standard review plan
TFSCS	Task Force[100] for Safety Critical Software
TMI	Three Mile Island
V&V	verification and validation

[99] A term used in [21] for a requirement on which the TFSCS has total consensus

[100] It consists of regulatory experts from the UK, Germany, Sweden, Belgium, Finland, and Spain

8 References

[1] US NRC Design-Specific Review Standard for the mPower Design, "Appendix A – Instrumentation and Controls: Hazard Analysis," ML12318A200, 2013.

[2] Corcoran, W.R., "Hazard recognition for quality, safety, and performance improvement," Special issue of the Firebird Forum, volume 15, number 3, March 2012.

[3] IEEE Standard 603-1991, "IEEE standard criteria for safety systems for nuclear power generating stations" 1991.

[4] MIL-STD-882E, "Standard Practice for System Safety," U.S. Department of Defense, 2012.

[5] Ericson II., C.A., "Hazard Analysis Primer," ISBN-13: 978-1470092535, 2012.

[6] U.S. Air Force, "The Air Force *System Safety Handbook"*, Kirtland AFB, NM, July 2000.

[7] National Aeronautics and Space Administration, "NASA Software Safety Guidebook*"*, NASA-GB-8719.13, Washington, DC, March 31, 2004.

[8] U.S. Nuclear Regulatory Commission, "Standard Review Plan for the Review of Safety Analysis Reports for Nuclear Power Plants: LWR Edition," Branch Technical Position 7-14, "Guidance on Software Reviews for Digital Computer-Based Instrumentation and Control Systems," NUREG-0800, Revision 5, Washington, DC, 2007.

[9] U.S. Nuclear Regulatory Commission, "Recommendations for Enhancing Reactor Safety in the 21st Century," The Near-Term Task Force Review of Insights from the Fukushima Dai-Chi Accident, Washington, DC, July 12, 2011.

[10] U.S. Nuclear Regulatory Commission, "Inadequate Flooding Protection Due to Ineffective Oversight," Licensee Event Report 285-2011-003, May 1, 2011.

[11] Garrett, C. and Apostolakis, G., "Context in the risk assessment of digital systems" Risk Analysis Vol. 19 No. 1 1999.

[12] U.S. Nuclear Regulatory Commission, Fault tree handbook (NUREG 492). URL: http://www.nrc.gov/reading-rm/doc-collections/nuregs/staff/sr0492/sr0492.pdf

[13] NASA, "Fault tree handbook with aerospace applications." URL: http://www.hq.nasa.gov/office/codeq/doctree/fthb.pdf

[14] SAE J1739, "Potential Failure Mode and Effects Analysis in Design (Design FMEA), Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes (Process FMEA), 2009" URL: http://standards.sae.org/j1739_200901/

[15] NASA, "Standard for Performing a Failure Mode and Effects Analysis (FMEA) and Establishing a Critical Items List (CIL) (DRAFT): Flight Assurance Procedure (FAP)-322-209," Nov. 2011, Available: rsdo.gsfc.nasa.gov/documents/Rapid-III-Documents/MAR-Reference/GSFC-FAP-322-208-FMEA-Draft.pdf

[16] Perrow, Charles, "Normal Accidents: Living with High Risk Technologies , New York: Basic Books, 1984"

[17] Proceedings of the NRC Advisory Committee for Reactor Safeguards 591st meeting, Rockville, Maryland, February 10, 2012. URL: http://pbadupws.nrc.gov/docs/ML1205/ML12054A637.pdf

[18] U.S. Nuclear Regulatory Commission, "Fort Calhoun Station – NRC Follow-up Inspection – Inspection Report 05000285/201007; Preliminary Substantial Finding," NRC Inspection Report 05000285/20010007, July 15, 2010.

[19] U.S. Nuclear Regulatory Commission, "Research Information Letter 1001: Software-related uncertainties in the assurance of digital safety systems Expert Clinic Findings, Part 1", ADAMS Accession Number ML1035402040, January, 2011.

[20] U.S. Federal Aviation Administration, "*System Safety Handbook"*, Washington, DC, December 2000.

[21] Task Force for Safety Critical Software, "Draft Licensing of Safety Critical Software for Nuclear Reactors," Common Position of Seven European Nuclear Regulators and Authorised Technical Support Organizations, Revision 2010. URL: http://www.hse.gov.uk/nuclear/software.pdf

[22] Steven P. Miller, Alan C. Tribble, Michael W. Whalen, and Mats P. E. Heimdahl. Proving the shalls: Early validation of requirements through formal methods. Int. J. Softw. Tools Technol. Transf., 8(4):303{319, 2006.

[23] Steven P. Miller, Michael W. Whalen, and Darren D. Cofer. Software model checking takes off. Commun. ACM, 53(2):58{64, 2010.

[24] PNO-77-146_8-19-77

[25] ISO/DIS 26262-2, "Road Vehicles – Functional Safety – Part 2: Management of functional safety", 2009.

[26] Garrett, C. and Apostolakis, G., "Automated hazard analysis of digital control systems" Reliability Engineering and Safety Society 77 (2002) 1-17

[27] NRC "Recent Operating Experience on Ineffective Use of Vendor Technical Recommendations" July 26, 2012, URL: http://nrr10.nrc.gov/forum/forumtopic.cfm?selectedForum=03&forumId=AllComm&topicId=3165&bookMark=3165201102101442456 73&searchId=1

[28] NRC IN-2012-11, "Age-related capacitor degradation" July 23, 2012 URL: http://pbadupws.nrc.gov/docs/ML1203/ML120330272.pdf

[29] P. Clements, R. Kazman, and M. Klein. Evaluating software architectures. Addison-Wesley, 2005

[30] ISO/IEC 25000: 2005(E) Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE

[31] CVE, see: http://cve.mitre.org/ (last accessed August 1st 2013)

[32] CWE, see: http://cwe.mitre.org/ (last accessed August 1st 2013)

[33] IEC 60880:2006 Nuclear power plants – Instrumentation and control systems important to safety – software aspects for computer-based systems performing category A functions.

[34] INTERNATIONAL NUCLEAR SAFETY ADVISORY GROUP, Defence in Depth in Nuclear Safety, INSAG-10, International Atomic Energy Agency, Vienna (1996).

[35] 10 Code of Federal Regulations (CFR) 50.34. URL: http://www.nrc.gov/reading-rm/doc-collections/cfr/part050/part050-0034.html

[36] 10 CFR 52.47(a)(2) URL: http://www.nrc.gov/reading-rm/doc-collections/cfr/part052/part052-0047.html

[37] Best Practices for Federal Research and Development Partnership Facilities. URL: https://www.ida.org/~/media/Corporate/Files/Publications/STPIPubs/2014/ida-p-5148.ashx

[38] Trustworthy Cyberspace: Strategic Plan for Cyber-security R&D Programs. URL: http://www.whitehouse.gov/sites/default/files/microsites/ostp/fed_cybersecurity_rd_strategic_plan_2011.pdf

[39] J.D. Lawrence, "NUREG/CR-6430 Software Safety Hazard Analysis," Lawrence Livermore National Labs, February 1996.

[40] S. Seth, et al, "NUREG/CR-6263 High Integrity Software for Nuclear Power Plants – Candidate guidelines, technical basis and research needs, Volume 1" The Mitre Corporation, June 1995.

Appendix A: Glossary

The scope of this glossary is limited to this document.

Where a word is not defined explicitly in the glossary, it is understood in terms of common usage as defined in published dictionaries of the English language (e.g., [1]).

The glossary focuses on terms that are not commonly understood in the same way, removing or reducing ambiguity by selecting and using more specific definitions. Where needed, notes elaborate the definition.

Where possible, the definition of a technical term is traceable to an authoritative reference source. In cases where the authorities have different, inconsistent definitions, the glossary adapts the definition and includes explanatory notes to reduce ambiguity.

The meanings of compound words, terms, and expressions are derived from the meanings of their constituent words, as defined in this glossary.

Aliasing

In signal processing and related disciplines, **aliasing** [1] refers to an effect that causes different signals to become indistinguishable (or *aliases* of one another) when sampled. It also refers to the distortion or artifact that results when the signal reconstructed from samples is different from the original continuous signal. (Also see anti-aliasing in [3]).

Assumption

A premise that is taken for granted (often implicitly), i.e., not validated.

Notes:

1. This definition is used in the context of reasoning as a part of safety analysis.
2. Other forms: Assume. Assumed. Assuming.

Analysis

A process of reasoning showing that a proposition can be deduced from premises (adapted from [2]).

Notes:

3. The process may entail decomposition. http://plato.stanford.edu/entries/analysis/s1.html#KD
4. See Kant's discussion at http://plato.stanford.edu/entries/analysis/s1.html#Kant
5. Analysis may take various forms:
    - 5.1. Quantitative
        - 5.1.1. Numerical (e.g., analysis of a continuous control algorithm)
        - 5.1.2. Logical
        - 5.1.3. Other forms of mathematical analysis. i.e.. where:
            - 5.1.3.1. The reasoning is composed with clear mathematical rules.
            - 5.1.3.2. The reasoning is backed by science (e.g., cause-effect laws of engineering).
    - 5.2. Qualitative[101], but consistently[102] repeatable across comparably qualified performers.

[101] See Quality

6. Performance of the analysis may entail various degrees of machine-assistance:
   6.1. Complete mechanization
   6.2. Mechanization, requiring manual interventional activities, e.g., human-guided machine-processing.
   6.3. Completely manual, but consistently repeatable across comparably qualified performers.
7. The term "formal" (along with its variations) is used to mean "mathematical" as in note 5.1.3
8. Derived forms:
   8.1. Analyzability
   8.2. Analyzable
   8.3. Un-analyzable
   8.4. Unanalyzed

Architecture

The structure or structures of the system, which comprise elements (e.g., software), the externally visible properties of those elements, and the relationships among them and with the environment (adapted from [5])

Where:

1. Externally visible properties of an element include behavior – normal, as well as abnormal – as seen from outside the boundary (interface) of an element.
2. Relationships include interactions and interconnections (communication paths).
3. Environment of the system includes the combination of systems and elements (e.g., hardware, software, and human) external to this system, human elements interacting directly with the system and the commensurate manual procedures.
4. System means combination of interacting elements organized to achieve one or more stated purposes. Systems can comprise of systems. A system with only software elements is also a system. For example, if a program comprises of subroutines, then the subroutines are elements and the program is a system.
5. In general, "ELEMENT" is a discrete part of a system that can be implemented to fulfill specified requirements. A system element can be hardware, software, data (structure), human, process (e.g., process for providing service to users), procedure (e.g., operator instructions), facility, materials, and naturally occurring entity (e.g., water, organism, mineral), or any combination.
   5.1. For a system (object of analysis) in the context of RIL-1101, "ELEMENT" can be hardware, software, or data (structure).

Assure

Confirm the certainty of correctness of the claim, based on evidence and reasoning.

Notes:

1. For example, by proof. For example, see note 5.1.3 in Analysis.
2. Examples of claims: (1) The system is safe (Property: Safety. Value: "Is safe.). (2) Property X of the system holds.

[102] If the analysis is not consistently repeatable or the analysis method/tool itself is not qualified for safe use, the purpose of this RIL treats the system as un-analyzable.

3. Derived forms:
   3.1. Assurance
   3.2. Assurable
   3.3. Assurability

Attribute (of quality)

Inherent property or characteristic of a system or its element that can be distinguished quantitatively or qualitatively. (Adapted from 2.2 in [33])

Notes:

1. The means of distinction may be manual or automated.
2. Also see "Quality measure" and "Scale."

Byzantine Behavior

In a distributed system, arbitrary behavior in response to a failure is called Byzantine behavior [6].

Note:
1. Arbitrary behavior of an element that results in disruption of the intended system behavior.
2. Different observers see different states.

Claim

A true-false statement about the value of a defined property of a system. (Adapted from [13])

Notes:

1. A property is a quality attribute of the system. (Adapted from 4.3.9 and 4.4.1 in [14])
   1.1. Example of property: Safety.
2. A property may have supporting sub-characteristics [14].
   2.1. Example: Verifiability ← Analyzability ← "Freedom from interference"
3. Unlike physical quantities, a property sub-characteristic may not be measurable on an absolute scale [14] .
   3.1. Indicators may be associated with a sub-characteristic for its estimation or indirect measurement.
4. A sub-characteristic may be specified in terms of conditions or constraints on its behavior [14] .
   4.1. Example sub-characteristic of safety property: Restriction on allowed system states.
   4.2. Example sub-characteristic of "Freedom from interference": Constraints on flows or interactions.
5. "Value" may be a single value, a set of single values, a range of values, a set of ranges of values, and limits on values. Value can be multi-dimensional [14].
6. "Value" may be invariant, dependent on time, or dependent on some other conditions [14].
7. Associated with a property may be the duration of its applicability (i.e., not limited to the present). For example, the property may concern the future behavior of the system [13].
8. Uncertainty (lack of certainty) may be associated with the property [13].
   8.1. The value of uncertainty may not necessarily depend upon probability..
   8.2. Uncertainty may be associated with a sub-characteristic.
   8.3. Uncertainty may be associated with the duration of applicability
   8.4. Uncertainty may be associated with other conditions of applicability
   8.5. For example, evaluation of a claim may be based upon certain conditions, formulated in terms of assumptions that the identified uncertainties do not exist.

Complexity

The degree to which a system or component has a design or implementation that is difficult to understand and verify. (Definition (1)(A) in [3])

Notes:

1. The selection[103] of this definition was favored by Dr. Gerard Holzmann [7].
2. The term, Simplicity, the converse of Complexity, is often used to discuss the same issues.
3. A "complexity measure or indicator" is often confused with the concept of "complexity", but should be distinguished as follows:
   - 3.1. A complexity measure pertains to any of a set of structure-based metrics that measure the attribute in Definition (1)(A) in [3]. (Definition (1)(B) in [3])
   - 3.2. Example of an indicator: The number of linearly independent paths (one plus the number of conditions) through the source code of a computer program is an indicator of control flow complexity, known as McCabe's cyclomatic complexity [3].
   - 3.3. Sometimes, the term "size-complexity" is used to refer to the effect of the number of states and number of inputs and their values and combinations.
4. Complexity theory is concerned with the study of the intrinsic complexity of computational tasks, that is, a typical Complexity theoretic study considers the computational resources required to solve a computational task (or a class of such tasks); it studies what can be achieved within limited time (and/or other limited natural computational resources) [8]. For example, the time required to solve a problem – calculated as function *f(…)* of the size of the instance, usually the size of the input, *n* – is studied for its scalability (e.g., bounded by "order of" *O(…)* with respect to the input size *n*). Similarly, instead of time, one could study the scalability with respect to some other resource constraint (e.g., space or memory). An example of a useful result from this theory is a premise that only those problems that can be solved in polynomial time, denoted as $O(n^k)$ for some constant *k*, can be feasibly computed on some computational device [9]. Applying this thesis to evaluation of system architecture, one could conclude that, if the input space of a system is not bounded, the system is not verifiable. One could further conclude that, if the interactions across elements of the system are not bounded, the system is not verifiable.

Complex Logic

An item of logic for which it is not practicable to ensure the correctness of all behaviors[104] through verification alone.

Notes:

1. This definition is derived from a combination of the definition of complexity given above and the following definition in DO-254/ED-80 in Appendix C [11], for "simple hardware item": "A hardware item is considered simple if a comprehensive combination of deterministic tests and analyses can ensure correct functional performance under all foreseeable operating conditions with no anomalous behavior." The conditional clause "if a comprehensive combination of deterministic tests and analyses…" is summarized as "verification."
2. Therefore, in addition to verification, the demonstration of correctness of Complex Logic requires a combination of evidence from various phases of the development life cycle, integrated with reasoning to justify the completeness of coverage provided (summarized as development assurance). Examples include the following:
   - 1.1. Evaluation of the system concept (and conceptual architecture)
   - 1.2. Evaluation of the verification and validation plan
   - 1.3. Criticality analysis

---

[103] Various standards provide different definitions; there is no broadly accepted definition.

[104] This refers to behaviour under all foreseeable operating conditions with no anomalous behaviour.

1.4. Evaluation of the architecture including requirements allocation
1.5. Evaluation of the system-internal hazard analysis
1.6. Validation of requirements and constraints on the design and implementation
1.7. Assessment and audit of all processes, including supporting and management processes.
1.8. Certifying[105] organizations developing software
1.9. Evaluation of the independence[106] of the assurance activities
1.10. See [11] for more detail.

3. Complex Logic is typically produced by techniques such as software or hardware description languages and their related tools. Thus, the assurance of correctness also requires commensurate assurance of the languages and tools.

Constraint

An externally imposed limitation on system requirements, design, or implementation or on the process used to develop or modify a system (Definition 6 in [31]).

Examples:

1. Pre and post conditions
2. Limits on memory size, cost, deadlines to be met.

Contribute

To play a significant part in bringing about an end or result (Definition 1b for contribute in [4])

Notes:

3. Derived forms:
   3.1. Contribution: The thing contributed
   3.2. **Contributory:** Of, relating to, or forming a contribution
4. Some experts use the term, "cause." Others sometimes interpret "cause" to mean "direct cause" or "primary cause" or "closely-coupled cause." However, many factors that influence the result may be distantly-coupled through long chains of dependency relationships; the term, "contribute" provides for their inclusion.

Defect

An imperfection or deficiency in a project component where that component does not meet its requirements or specifications and needs to be either repaired or replaced*. A Guide to the Project Management Body of Knowledge (PMBOK® Guide) — Fourth Edition.* [31]

Notes:

1. The condition "that component does not meet its requirements or specifications" would exclude cases where the requirement or specification itself is deficient.
2. Another definition in [31] "a problem which, if not corrected, could cause an application to either fail or to produce incorrect results*. ISO/IEC 20926:2003, Software engineering — IFPUG 4.1 Unadjusted functional size measurement method — Counting practices manual*" depends upon the definition of "failure" and "correctness" both of which, in turn, are evaluated with respect to requirements. Thus, this definition would also exclude cases where the requirement or specification itself is deficient.

---

[105] Certification of the development organization should be a continual process of certification and recertification much in the same manner as reactor operators are certified periodically. For example, the capability maturity model integrated certification process developed by the Software Engineering Institute focuses on assessing the capabilities of development.

[106] For example, independence can be evaluated through certification of the assurance process for the Complex Logic (e.g., software).

3. From notes 1 and 2, it can be seen that a system may not be defective; yet it may lead to a hazard.
4. In RIL-1101, the term is used primarily in the context of the engineering phases of the product lifecycle.

Demonstrate

Prove (the assertion in context) through reasoning, connecting evidence.

Dependent

Determined or conditioned by another.

Notes:

1. Other forms:
   1.1. Dependence; Dependency: The quality or state of being dependent upon or unduly subject to the influence of another.
   1.2. Independent
2. .

   2.1. The quality or state of being dependent upon or unduly subject to the influence of another.

Diverse team

A team composed of individuals with complementary attributes needed to perform the assigned task (e.g., thought processes, communication styles, and competence, including education training, and experience in different domains and disciplines).

(System) Element

A discrete constituent of a system (adapted from [16]).

Notes:

3. The term “discrete constituent” is substituted for the word “component” used in the definition from [16]. Reason: Avoid confusion with other meanings of “component” in the context of software. The word “discrete” implies that the constituent has a distinct boundary, that is, interface with its environment (per definition in [17]), and an intrinsic, immutable, unique identity (adapted from [16]).
4. Examples:
   4.1. Hardware element
   4.2. Software element
   4.3. Human element
   4.4. Data element
   4.5. Process
   4.6. Procedure (e.g., operating instructions)
5. An element may have other elements in it (e.g., a subsystem).
6. A system may itself be an element of a larger system.

Environment

A general term relating to everything (including every condition) that supports or affects the performance of a system or a function of the system. (Combination of 9A and 9B in [3] which refer to (C) 610.12-1990)

Notes:

1. The environment of a software component consists of all the elements (in their respective states or conditions), with which it interacts, by which it is affected, and on which it depends. Examples of elements:
   1.1. Other software components
   1.2. Operating system (common services and resources shared by software components)
   1.3. Execution hardware
2. The environment of an electronic hardware component consists of physical environmental conditions and other hardware components (in their respective states or conditions) with which it interacts, by which it is affected, and on which it depends. Examples of physical environmental conditions:
   2.1. Temperature
   2.2. Humidity
   2.3. Electromagnetic radiation

Error

The difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition (Definition (8)(A) in [3])

Evidence

Data supporting the existence or verity of something. (Adapted from 3.1936 in [31])

Note:

1. Examples of means of obtaining “raw” evidence: Test; measurement; observation.
2. Examples of evidence incorporating reasoning:
   2.1. Confirmation by static analysis that an implementation satisfies its design specification.
   2.2. A claim at one level of integration used as evidence in claim for next higher level of integration of a system.

Failure

The termination of the ability of an item to perform a required function. [18]

Notes:

1. After failure, the item has a fault. [18]
2. “Failure” is an event, as distinguished from “fault” which is a state. [18]
3. This concept as defined does not apply to items consisting of software only.[18]
4. The following definitions represent the perspectives of different disciplines to reinforce the definition given above:
   4.1. The termination of the ability of an item to perform a required function (Definition (1)(A) in [3]).
   4.2. The termination of the ability of a functional unit to perform its required function (Definition (1)(N) in [3]).
   4.3. An event in which a system or system component does not perform a required function within specified limits; a failure may be produced when a fault is encountered (Definition (1)(O) in [3]).
   4.4. The termination of the ability of an item to perform its required function (Definition 9 in [3] from “nuclear power generating station”).
   4.5. The loss of ability of a component, equipment, or system to perform a required function (Definition 13 in [3] Safety systems equipment in “nuclear power generating stations”).
   4.6. An event that may limit the capability of equipment or a system to perform its function(s) (Definition 14 in [3] “Supervisory control, data acquisition, and automatic control”).

4.7. The termination of the ability of an item to perform a required function (Definition 15 in [3] “nuclear power generating systems”)

Failure Analysis

The logical, systematic examination of a failed item to identify and analyze the failure mechanism, the failure cause, and the consequences of failure. (191-16-12 in [18])

Fault

The state of an item characterized by inability to perform a required function, excluding the inability during preventive maintenance or other planned actions, or due to lack of external resources. (191-05-01 in [18])

Notes

1. A fault is often the result of a failure of the item itself but may exist without prior failure.
2. Also see “defect.”
3. Distinguish from failure, mistake, and error.
4. (Derived form) Faulty: Pertaining to an item that has a fault.

Fault Analysis

The logical, systematic examination of an item to identify and analyze the probability, causes, and consequences of potential faults. (191-16-11 in [18])

Fault Mode

One of the possible states of a faulty item, for a given required function.

Note:
RIL-1101 does not use the term “failure mode” in this sense.

Fault Modes and Effects Analysis (FMEA)

A qualitative method of reliability analysis, which involves the study of the fault modes, which can exist in every sub-item of the item, and the determination of the effects of each fault mode on other sub-items of the item and on the required functions of the item. (191-16-03 in [18])

Note:
RIL-1101 does not use the term “failure mode and effects analysis” in this sense.

Fault tolerance

The ability of a system or component to continue normal operation despite the presence of hardware or software faults (Definition 1 in 3.1127 in [31]).

Notes:

1. “Fault tolerance” is also defined as a discipline pertaining to the study of errors, faults, and failures, and of methods for enabling systems to continue normal operation in the presence of faults (Definition 3 in 3.1127 in [31]).
2. Derived forms “Fault tolerant”, “Fault-tolerant”: pertaining to a system or component that is able to continue normal operation despite the presence of faults (3.1128 in [31]).
3. For example: Conditions that may degrade the performance of a function of the system are identified; in anticipation, a constraint is formulated to prevent such degradation; and the resulting system is able to continue performance of the required function when the anticipated conditions arise.

Fault Tree Analysis (FTA)

An analysis to determine which fault modes of the sub items or external events, or combinations thereof, may result in a stated fault mode of the item, presented in the form of a fault tree. (191-16-05 in [18]).

Feasible

Capable of being done with the means at hand and circumstances as they are. [20]

Notes:

1. Other definitions also impose such constraints as
   1.1. Practicably
   1.2. Reasonable amount of effort, cost, or other hardship [21]
   1.3. Cost-effectiveness. [22]
2. Such constraints distinguish “feasibility” from “possibility.”

Freedom from interference

Freedom from degradation of the performance of a function due to interaction across the system and its environment or across elements of the system.

Note:

1. Interference: Interaction across a system and its environment or across elements of a system that can degrade the performance of a function. It is not limited to propagation of a failure.

Hardwired

Pertaining to a circuit or device whose characteristics and functionality are permanently determined by the interconnections[107] between components[108] (Adapted from Definition 3 in [3]).

Note:

The referred-to connections are at the printed circuit board level (or cabinet level), not internal to integrated circuits.

Hazard

Potential for harm[109]

Examples:

1. A condition;
2. A circumstance;
3. A scenario.

Notes:

1. RIL-1101 bounds the scope to the entity (system; element) in the context of a defined environment.
2. At the initial stage of hazard logging (before any analysis of the initial finding), the log may include an item, which, after some analysis, is re-characterized (differently from the originally characterized hazard; possibly, an event).
3. Definition A in [15] (same as definition 3.1283-1 in [31]) elaborate on the “potential for harm” as follows, “An intrinsic property or condition that has the potential to cause harm or damage.”

Contributory hazard

[107] Examples: Wiring in cabinets; Printed paths in circuit boards

[108] Examples: Relays; AND-gates; OR-gates

[109] In general, “loss” of any kind that is of concern. Focus of RIL-1101: Harm.

Factor contributing to potential for harm.

Notes:

1. (Excerpt from [23]) …. An unsafe act and / or unsafe condition which contributes to the accident[110], ...
2. Figures 7-1 - 7-4 in [24] illustrate contribution paths.

Examples:

1. The potential for adverse energy flow [23]
2. Inappropriate functions (from Figure 7-5 in [24])
3. Normal functions that are out of sequence (from Figure 7-5 in [24])
4. Functional damage and system degradation (from Section 7.1.1 in [24])
5. Machine-environment interactions resulting from change or deviation stresses as they occur in time and space (from Section 7.1.1 in [24])

Hazard Analysis

Hazard analysis (HA) is the process of examining a system throughout its lifecycle to identify inherent hazards (see) and contributory hazards, and requirements and constraints to eliminate, prevent, or otherwise control them.

Notes:

1. "Hazard identification" part of HA includes the identification of losses (harm) of concern.
2. This definition is narrower than many definitions of HA, some of which correspond to the NRC's usage of the term "safety analysis" (as in a safety analysis report).
    a. The scope of the definition excludes the verification that the requirements and constraints have been satisfied.
    b. Various HA definitions and descriptions identify artifacts (results, including intermediate results) of HA by different names. The expression "requirements and constraints" used in this definition (to align and integrate them in well-established systems engineering terms) subsumes them.
    c. The scope of the definition does not include quantification explicitly. Where appropriate (e.g., for a hardware component, quantification of its reliability would be implicit in the activity of formulating requirements and constraints).

Hazard Identification

The process of recognizing that a hazard exists and defining its characteristics [31].

Indicate

To be a sign, symptom, or index of [1].

Note:

1. Derived form: **Indicator** – A device or variable that can be set to a prescribed state based on the results of a process or the occurrence of a specified condition. [3]
2. Often an indicator is an estimate or a result of evaluation, possibly incorporating judgment, and not measured on a standardized scale (or norm).
3. An indicator is created for its potential utility by facilitating comparison of current state with goal state, rather than for absolute accuracy.
4. Contrast with quality measure.

[110] in our case, degradation of a safety function

Intended

Intentional (Meaning 2 in [4])

Notes

1. Derived form: Unintended; meaning "not intentional," i.e., not required directly or indirectly.
2. Also see

Information hiding

The principle of segregation of design decisions in a computer program that is most likely to change, thus protecting other parts of the program from extensive modification if the design decision is changed. The protection involves providing a stable interface which protects the remainder of the program from the implementation (the details that are most likely to change).

Item (Entity)

Any part, component, device, subsystem, functional unit, equipment, or system that can be individually considered. (191-01-01 in [18])

Notes:

1. In [15], The term, element, is used to mean item.
2. An item may consist of hardware, software, or both, and may, in particular cases, include people.
3. A number of items (e.g.,, a population of items) or a sample may itself be considered an item.

Mechanize

to produce by or as if by machine [4].

Mistake

A human action that produces an unintended result (Definition 1 in [3] "electronic computation")

Editorial note (contrary to the note attached to Definition 1 in [3]): In the context of software engineering, this definition should be applied to mistakes concerning requirements development (including elicitation, transformation of intent into requirement or constraint specification, and explicit statement of assumptions (e.g., about the environment) and respective validation.

A human action that produces an incorrect result (Definition 3 in [3] "software")

Note: The fault tolerance discipline distinguishes between the human action (a mistake), its manifestation (a hardware or software fault), the result of the fault (a failure), and the amount by which the result is incorrect (the error). [3]

Editorial note (complementing the note in the previous definition of "mistake"): In the context of software engineering, this definition should be applied to mistakes concerning transformation of requirements specifications and constraints into successive work products and their respective verification.

Mode confusion

A situation in which an engineered system can behave differently from its user's expectation, because of a misunderstanding or inadequate understanding of the system state.

Process

A set of interrelated activities, which transforms inputs into outputs. (Definition 12(A) in [3]. Definition 3.2217-1 in [31])

Notes

1. Definition 4 in [3] makes “including the transition criteria for progressing from one (activity) to the next” explicit.
2. In definition 4 in [3], the expression “that bring about a result” corresponds to “which transforms inputs into outputs.” The latter is used in the definition above, because it identifies a set of starting conditions (inputs), a set of end conditions (outputs) and the transformational purpose of the process.
3. Examples of transformational processes in an engineering lifecycle of a product: Requirements; Architecture; Detailed design; Implementation. If the overall engineering is considered a lifecycle process, then these may be identified as phases in that lifecycle process.

Product

Result of a process. (3.2257-4 in [31])

Notes:

1. Referring to Note 3 for process, the term “product” may be used for the final product or for a result of a particular phase of a lifecycle process; for example: System requirements specification; System architecture specification; Detailed design specification; (Software) source code; (Software) executable code.

Quality

Capability of product to satisfy stated and implied needs when used under specified conditions. (Adapted from 4.51 in [32])

Notes

1. This definition differs from the ISO 9000:2000 quality definition; it refers to the satisfaction of stated and implied needs, while the ISO 9000 quality definition refers to the satisfaction of requirements.
2. The term “implied needs” means “needs that may not have been stated explicitly (e.g., a need that is considered to be evident or obvious; a need implied by another stated need).”
3. **Quality model**: Defined set of characteristics, and of relationships between them, which provides a framework for specifying quality requirements and evaluating quality. (Adapted from 4.44 in [32])
4. **Quality measure**: An attribute of quality to which a value is assigned. Also see scale.
5. **Quality in use**: Capability of the product to enable specific users to achieve specific goals in specific contexts of use. The expression “in use” refers to the expectations of the end user.
    5.1. Actual quality in use may be different from quality in use measured in a test environment earlier in the product lifecycle, because the actual needs of users may not be the same as those reflected in the test cases or in the requirements specifications.
    5.2. Quality in use requirements contribute to identification and definition of external software quality requirements.
    5.3. Example of quality in use: Safety (freedom from harm).
6. Measurement of external quality refers to measurement from an external view of the product, where targets are derived from the expected “quality in use” and are used for technical verification and validation. For example, external software quality would be measured in terms of its capability to enable the behavior of the system to satisfy its quality in use requirements, such as safety.
7. Measurement of internal quality refers to measurements during the developmental phases of the product lifecycle. Targets are derived from targets for measurement of external quality.

Reason

Argument: A logical sequence or series of statements from a premise to a conclusion. (Adapted from http://www.merriam-webster.com/dictionary/argument. Also see

Notes:

1. **Argument:** Also see http://www-rohan.sdsu.edu/~digger/305/toulmin_model.htm
2. Derived forms:
   2.1. **Reasoning**: The use of reason
   2.2. **Reasonable**: Being in accordance with reason. (http://www.merriam-webster.com/dictionary/reasonable)

Reliability (symbol : $R(t_1, t_2)$)

The probability that an item can perform a required function under given conditions for a given time interval (t_1, t_2). (191-12-01 in [18])

Notes:

1. It is generally assumed that the item is in a state to perform this required function at the beginning of the time interval.[111]
2. The term "reliability" is also used to denote the reliability performance quantified by this probability (see 191-02-06 in [18]).
3. This definition does not apply to items for which development mistakes can cause failures, because there is no recognized way to assign a probability to development mistakes.

Requirement

Expression of a perceived need that something be accomplished or realized. (Adapted from 4.47 in [32])

Notes:

1. Functional requirement: Requirement that specifies a function that a system or its element must be able to perform, (Adapted from 4.22 in [32])
2. Quality requirement: Requirement that specifies a quality of a system or its element, where quality may be one of the following:
   2.1. Quality in use (e.g., safety). Quality in use requirements specify the required level of quality from the end user's point of view. Also see note 5 in definition of quality.
   2.2. External quality. Also see note 6 in definition of quality.
   2.3. Internal quality. Also see note 7 in definition of quality.

Resilience

The property of a system or its element to recover from fault.

Notes:

1. "Resilience" as used in this context is not defined in any of the standards used as references for safety, systems, or software engineering. This usage is metaphoric. derived from the common usage meanings given in notes 2-3. Use the term "Fault tolerance" usage of which is well supported in the fault tolerance discipline.
2. "Resilience" is most commonly used and defined in the context of people. For example: Resilience is the capacity to withstand stress and catastrophe. (http://www.pbs.org/thisemotionallife/topic/resilience/what-resilience)
3. "Resilience" is also used and defined as a mechanical property of an object or material. For example: The physical property of a material that can return to its original shape or position after deformation that does not exceed its elastic limit. (http://www.webster-dictionary.org/definition/resilience)

---

[111] For a software component that is faulty to begin with, use of the term reliability is neither meaningful nor helpful; instead, it leads to the misapplication of analysis techniques that served well for traditional hardware.

Robustness

The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions. (3.2601 in [31])

Scale (for a quality measure)

Ordered set of values, continuous or discrete, or a set of categories to which an attribute is mapped. (Adapted from 2.35 in [33])

Notes

1. The type of scale depends on the nature of the relationship between values on the scale [33].
2. Four types[112] of scale are commonly defined [33]:
   2.1. Nominal: The measurement values are categorical
   2.2. Ordinal: The measurement values are rankings
   2.3. Interval: The measurement values are equi-spaced
   2.4. Ratio: The measurement values are equi-spaced, where the value 0 (zero) is not mapped to any attribute.
3. The valid value space is predetermined.
4. The mapping of the magnitude of the measured attribute to a value on the scale is predetermined.

Separation of concerns

The process of separating a computer program into distinct features that overlap in functionality as little as possible. A concern is any piece of interest or focus in a program. Typically, concerns are synonymous with features or behaviors. [25]

State

The present condition of a (dynamic) system or entity.

Note:

A state is a complete set of observable properties (also known as state variables) that characterize the behavior of a system, that is, response to stimuli (set of inputs).

State space

The set of all possible states of a dynamic system [26].

Note:

Each state of the system corresponds to a unique point in the state space.

System

Combination of interacting elements organized to achieve one or more stated purposes [27].

Notes

1. A system may be considered as a product or as the services it provides (adapted from [27]). For example, at its conceptualization stage, a system may be described in terms of the services it provides and its interactions with its environment, without identifying its constituent elements.
2. The expression “combination…organized…” (instead of collection) emphasizes that a system is an “integrated composite” as characterized from the definition in [28] of system.

[112] See [34] for other types of scale.

3. The expression “to achieve its stated purposes” corresponds to the expression “a capability to satisfy a stated need or objective” used in the definition in [28] of system.
4. In practice, the interpretation of its meaning is frequently clarified by the use of an associated noun (e.g., reactor protection system). (Adapted from [27])
5. System elements may include people, products and processes (adapted from [28]). In the boundary of NRC’s licensing review plan (DSRS) Chapter 7, the review of a digital safety system is focused on the safety automation. Operators, thermo-hydraulic processes, and related supporting, peripheral processes are part of the environment of the digital safety system. The scope of Chapter 7 review includes Interactions of the digital safety system with its environment.

Systemic

Embedded within and spread throughout and affecting a group, system, or body.

Systematic Failure

Failure, related in a deterministic way to a certain cause, that can be eliminated only by a modification of the design or of the manufacturing process, operational procedures, documentation, or other relevant factors. [18]

Notes

1. Corrective maintenance without modification will usually not eliminate the failure cause.
2. A systematic failure can be induced by simulating the failure cause.
3. In International Electrotechnical Commission 61508-4 CDV 3.6.6 [30]: Examples of causes of systematic failures include human mistakes in the following areas:
    a. The safety requirements specification
    b. The design, manufacture, installation, and operation of the hardware
    c. The design, implementation, etc. of the software
4. Other examples include mistakes in modification and configuration.
5. Also, see “systemic cause” in [29].

Traceability

Discernible association among two or more logical entities, such as requirements, system elements, verifications, or tasks.

Unwanted

Not needed (Derived from Definition 3 for want in [4])

Notes

1. The need is not intrinsic to the specified requirements.

Validation

Confirmation that a product satisfies the needs of the customer and other identified stakeholders. (Adapted from 3.3264-5 in [31]).

Notes

2. “Confirmation” is used instead of “Assurance,” the word used in [31]. Rationale:
    2.1. Avoid confusion with the use of the word “Assurance” in RIL_1101.
    2.2. Consistency with the use of “Confirmation” in the definition of “Verification.”
    2.3. “Confirmation” subsumes the term, “the process of evaluating” used within definition A in [15].
    2.4. “Confirmation” subsumes the term, “the process of providing evidence” used within definition B in [15].
3. “Validation” includes confirmation that the requirements are correct, complete, consistent, and unambiguous.

4. The stakeholder requirements definition activity includes the transformation of various needs into requirements, including the requirements for validation [10].
   4.1. In [15], validation of stakeholder requirements definition includes HA.
   4.2. In the context of an NPP safety system, "stakeholder requirements" mean NPP safety requirements allocated to and intended for this safety system.
   4.3. "Requirements for validation" include Assurability.
5. The activity of validation includes the confirmation that the specification for each lifecycle phase satisfies the needs of the customer and other identified stakeholders.
6. A clarification of the expression, "the needs of the customer and other identified stakeholders" is provided within definition B in [15] as follows: Solve the right problem (e.g., correctly model physical laws, implement business rules, and use the proper system assumptions), and satisfy intended use and user needs.
7. The concept of "validation," as defined, subsumes the concept of "verification." However, there is a lack of clear agreement across various authorities on the subsumption of "verification" in "validation."
8. "Product" subsumes the elaboration, "system, software, or hardware and its associated products" used within definition B in [15].
9. "Satisfies" is used instead of "meets," the word used in [31]. Rationale: Consistency with usage in the definition of "Verification."
10. The elaboration "….satisfy requirements allocated to it at the end of each life cycle activity" within definition B in [15] is subsumed in the expression, "satisfies the needs of the customer and other identified stakeholders".

Verification

Confirmation that specified requirements have been satisfied. (Adapted from **3.3282-3 in** [31]).

Notes

1. Various standards and authorities have different definitions, which are inconsistent with each other. The definition given above abstracts commonality to the extent possible. The following notes provide explanations, with attempts to reconcile some differences across certain definitions where possible.
2. The term is also used to mean the process of confirmation that specified requirements have been satisfied. The usage context will distinguish the two meanings.
   2.1. Definition A in [15] characterizes the verification process "… evaluating … to determine whether … product …. satisfy … " If the result of the determination is TRUE, then it is "confirmation." The act of evaluating includes reviewing, inspecting, testing, checking, auditing, or otherwise determining and documenting (also see note 9 below).
   2.2. The object of verification is implied in the definition (e.g., confirmation that a product satisfies its specified requirements).
3. Definition 3 in [31] uses the term "fulfilled"; however, to reduce potential ambiguity, the term "satisfied" is used (which is also used in definition 1 within [31]) in the general sense of propositional satisfaction (⊨) and constraint satisfaction.
   3.1. Definition 2 in [31] uses the term "formal proof" favoring this substitution.
   3.2. Definition 6 in [31] uses the term "comply with" which may be mapped conservatively into "satisfies."
   3.3. Definition B in [15] uses the term "conforms to" which may be mapped conservatively into "satisfies."
4. Definitions 3 and 6 in [31] also include the phrase "through the provision of objective evidence." This phrase is omitted, because the concept "satisfied," as explained in Note 3 subsumes it,
5. Definition A in [15] uses the expression "satisfy the conditions imposed at the start of that phase"; this expression is mapped into "specified requirements" in the definition above.
6. Definition B in [15] elaborates "… for all life cycle activities during each life cycle process"; the definitions of product and process subsume this elaboration.
7. Definition B in [15] elaborates "satisfy standards, practices, and conventions during life cycle processes; and successfully complete each life cycle activity and satisfy all the criteria for initiating succeeding life cycle

activities"; the term "specified requirements" in conjunction with definitions of product and process subsumes this elaboration.

8. Definition B in [15] includes the statement "Verification of interim work products is essential for proper understanding and assessment of the life cycle phase product(s)." This statement does not add to the definition of verification.
9. Definition 3 in [3] elaborates "The act of reviewing, inspecting, testing, checking, auditing, or otherwise determining and documenting whether …"; the term "process" in the definition given in Note 2 abstracts this elaboration.
10. Verification at each lifecycle phase does not imply verification of the end product, because its scope does not include the confirmation that the specification for each lifecycle phase satisfies the requirements at the initial phase (e.g.,, stakeholder requirements [15] for the end product). This confirmation is considered a part of validation activities; however, there is a lack of clear agreement across various standards and authorities on this separation of verification and validation.

References for Appendix A

[1] Wikipedia.org, "Aliasing," <http://en.wikipedia.org/wiki/Aliasing>, October 16, 2012.

[2] Caygill Howard, *A Kant Dictionary*, 1995 .

[3] Institute of Electrical and Electronics Engineers, "The Authoritative Dictionary of IEEE Standards Terms," IEEE Standard 100-2000, 7th edition, 2000.

[4] Merriam-Webstert.com,"<http://www.merriam-webster.com/dictionary/>, 2014.

[5] Bass, Clements, Katzman, "Software Architecture in Practice (2nd edition)" Addision-Wesley 2003; quoted at the URL: http://www.sei.cmu.edu/architecture/start/glossary/moderndefs.cfm

[6] Schneider, F., "Understanding protocols for Byzantine clock synchronization" Dept of Computer Science, Cornell University, Ithaca, New York, Technical Report # 87-859, August 1987.

[7] U.S. Nuclear Regulatory Commission, "Research Information Letter 1001: Software-related uncertainties in the assurance of digital safety systems Expert Clinic Findings, Part 1", ADAMS Accession Number ML1035402040, January, 2011.

[8] Goldreich, Obed, "Computational Complexity: A Conceptual Perspective," ISBN 978-0-521-88473-0, Cambridge University Press, May 2008.

[9] Cobham, Alan, "The intrinsic computational difficulty of functions", Proc. Logic, Methodology, and Philosophy of Science II, North Holland, 1965.

[10] ISO/IEC/IEEE 15288 Systems and Software Engineering—System Life Cycle Processes

[11] RTCA DO-254/Eurocae ED-80 Standard, "Design Assurance for Airborne Electronic Hardware," Radio Technical Commission for Aeronautics/EUROCAE, April 19, 2000.

[12] Merriam-Webstert.com, "Contribute,"<http://www.merriam-webster.com/dictionary/contribute>, October 15, 2012.

[13] ISO/IEC TR 15026-1:2010 Systems and software engineering – Systems and software assurance – Part 1: Concepts and vocabulary, revised as ISO/IEC DIS 15026-1:2013

[14] ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.

[15] Institute of Electrical and Electronics Engineers, "IEEE Standard for System and Software Verification and Validation," IEEE Standard 1012-2012, IEEE Computer Society, 2012.

[16] Institute of Electrical and Electronics Engineers, “IEEE Guide for Developing System Requirements Specifications,” IEEE Standard 1233-1998, 1998.

[17] International Electrotechnical Commission, “Information Technology – Vocabulary – Part 1: Fundamental Terms,” ISO/IEC 2382-1:1993, 1993.

[18] International Electrotechnical Commission, “International Electrotechnical Vocabulary, Chapter 191: Dependability and Quality of Service,” IEC 60050-191:1990-12, 1st edition, 1990.

[19] BusinessDictionary.com, “Design Defect,” <http://www.businessdictionary.com/definition/design-defect.html>, December 17, 2010.

[20] WordNet, “Feasible,” Princeton University, <http://wordnetweb.princeton.edu/perl/webwn?s=feasible>, December 17, 2010.

[21] U.S. Department of Transportation, Federal Highway Administration, “Feasible,” <http://www.fhwa.dot.gov/environment/sidewalks/appb.htm>, December 17, 2010.

[22] Georgetown University, “Feasible,” <http://uis.georgetown.edu/departments/eets/dw/GLOSSARY0816.html>, December 17, 2010.

[23] AviationGlossary.com, “Contributory Hazard,” <http://aviationglossary.com/aviation-safety-terms/contributory-hazard/>, October 15, 2012.

[24] FAA System Safety Handbook, Chapter 7: Integrated System Hazard Analysis, December 30, 2000.

[25] Wikipedia.org, “Separation of Concerns,” <en.wikipedia.org/wiki/Separation_of_concerns>, October 15, 2012.

[26] Scholarpedia.org, “State Space,” <http://www.scholarpedia.org/article/State_space>, October 15, 2012.

[27] Institute of Electrical and Electronics Engineers, “Systems and Software Engineering – Software Life Cycle Processes,” IEEE Standard 12207-2008, 2008.

[28] U.S. Department of Defense, “Standard Practice for System Safety,” MIL-STD-882E, May 11, 2012.

[29] International Organization for Standardization, “Road Vehicles—Functional Safety—Part 1: Vocabulary,” ISO/DIS 26262-1, 1st edition, 2009.

[30] Chris Eckert, Apollo Associated Services, LLC, “Identification and Elimination of Systemic Problems,” Proceedings of the Society of Maintenance and Reliability Professionals Annual Symposium, St. Louis, MO, October 20–22, 2009.

[31] ISO/IEC/IEEE 24765 Systems and software engineering – vocabulary, 2010

[32] ISO/IEC 25000: 2005(E) Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE

[33] ISO/IEC 15939:2007(E) Systems and software engineering – Measurement process

[34] Roberts, F. Measurement Theory with Applications to Decision Making, Utility, and the Social Sciences, Addison-Wesley, 1979

Appendix B: Technical Review Process

Technical reviews were performed iteratively reviews with the purpose of acquiring knowledge outside the nuclear power plant (NPP) domain relevant to evaluation of an applicant's hazard analysis (HA) of a digital instrumentation and control (DI&C) system for safety functions in a NPP.

The Office of Nuclear Regulatory Research (RES) employed the services of Safeware Engineering Corporation (SEC) [1] as a neutral agent to interface with external experts. SEC obtained nine experts spread across safety-critical software and systems research experience outside of the commercial NPP industry (e.g., space exploration, military defense, aviation industry).

Unlike typical peer reviews, in this process, the expert provided the content needed to bring the report to the expert's standard of technical soundness, along with an explanation and justification of the modification, addition or subtraction.

Review process

The technical reviews were performed iteratively at evolving stages of RIL-1101. Each iteration was treated as a knowledge-acquisition cycle from which results were integrated into the development of RIL-1101, before submitting it for the next review cycle.

Each review cycle followed the procedure outlined below:

1. NRC and SEC provided orientation to the expert as follows
   1.1. NRC sent to the expert three documents to prepare for a face-to-face discussion:
      1.1.1. A draft of RIL-1101
      1.1.2. A review template specific to the review cycle
      1.1.3. A set of slides introducing the NPP application domain, key issues addressed in RIL-1101, and scope and request-response sequence for the project.
   1.2. Then, in a face-to-face meeting, NRC and SEC walked the expert through the slide set, engaging the expert in clarifying discussion. Then, NRC and the expert discussed the review template for clarification of the task and match of expert's interest. The review was scoped accordingly.
2. The expert provided a written review response as follows:
   2.1. Responses to specific questions in the NRC-provided review template. Typically, the expert provided these responses in tabular form as suggested in the template.
   2.2. Rationale or explanation supporting the proposed changes;
   2.3. Supporting references, mostly incorporated by reference;
   2.4. Supporting examples or case studies in the expert's experience or research to support an assertion or guidance item applicable to the scope of RIL-1101 (e.g., through abduction or induction or other manner of generalization).
3. NRC staff and the expert discussed the expert's responses in a teleconference, moderated by SEC. Most of the responses concerned clarity of the intended messages. For "easy-to-resolve" comments, the disposition was discussed in the teleconference.

4. In some cases, the expert provided modified or supplemental responses.
5. NRC proposed disposition of expert's suggestions, sometimes including follow-up questions for discussion with the expert.
6. NRC discussed its proposed disposition with the expert. Depending on need and scheduling feasibility, sometimes NRC walked the expert through the disposition in a teleconference. In most cases, NRC met the expert face-to-face to clear remaining issues that could not be resolved efficiently through teleconferencing.

Although the initial plan had included resolution of conflicting inputs from different experts through cross-expert discussion, there was no conflict across experts about technical soundness.

References for Appendix B

[1] U.S. Nuclear Regulatory Commission, "Digital Instrumentation and Control – Technical Engineering Services," Statement of Work for Commercial - V6065, March 2012.

Appendix C: Evaluating Hazard Analysis - State of the Art

The scope[113] of this appendix is limited to the scope of RIL-1101, especially analysis of contributory[114] hazards in digital safety systems for NPPs, which are rooted in systemic causes. For example, it does not discuss techniques or aspects for analysis of systems with a mix of safety and non-safety functions (mixed-criticality systems) or analysis of hazards from random hardware failure. Whereas almost all the surveyed publications cover mixed-criticality systems, this appendix maps the extracted information into its narrower scope. For example:

1. Only a relevant subset of the wide range of HA activities is extracted.
2. The starting point of hazard analysis is "loss/degradation of an allocated safety function, rather than the unwanted release of radioactivity.

C.1 Reference model for hazard analysis: Vocabulary

The vocabulary in this appendix is defined in Appendix A. Following is an explanation of the usage context. A hazard is potential for harm, as defined in Appendix A. Bounding its context to the object of analysis and its environment, this definition is elaborated in its notes as follows, "an intrinsic property or condition that has the potential to cause harm or damage." In the scope of RIL-1101, the context of "the intrinsic condition" is a safety related system (or its element) being analyzed and its dependency on its environment. In other words, a hazard is a state[115] of the object of analysis together with its environment, which has the potential to cause harm. Hazard analysis (HA) of an object is the process of examining the object throughout its lifecycle to identify hazards (including contributory hazards), and requirements and constraints to eliminate, prevent, or otherwise control these hazards.

C.1.2 Object of analysis

Referring to the reference model for system integration levels depicted in Figure 4 of [1], the object of analysis may be any of the following:

1. A work product such as the following:
   1.1. A complete safety system such as a reactor protection system (RPS).
   1.2. One of its four identical divisions; (information source: system architecture).
   1.3. An element responsible for the voting logic; (information source: system architecture).
   1.4. A system at a lower level of integration; (information source: system architecture).
   1.5. The finest-grained component in the integration hierarchy; (information source: software architecture; hardware architecture).
   1.6. An object in the environment of the object being analyzed, on which the latter depends; (information source: NPP-wide I&C architecture).

[113] Thus, the definitions and descriptions are much more narrowly focused than in more broadly applicable publications on hazard analysis.

[114] IEEE1012-2012 [1] introduces the notion of contributory hazards, e.g. software and hardware contributions to system hazards.

[115] Annex J.1 in [1] "…determine whether the contributing conditions to a hazardous state are possible."

1.7. Result of an intermediate phase to produce any of the above; (information source: development lifecycle model).

2. A process activity producing a work product mentioned above; (information source: process activity model).
3. A resource used in a process activity mentioned above; (information source: process activity model). See in RIL-1101 Figure 4.
4. Any other object in a path of contributory hazards.

C.1.3 Analysis at different levels in the dependency network

The dependency network of the top-level system provides an organizing framework for these objects. For each object, the starting point of its HA would correspond to the derived requirements assigned to it, its boundary with respect to its environment, its relationship to its environment, and associated assumptions. If HA of different objects is occurring concurrently (e.g., impact of changes), based on assumptions about their place and relationships in the dependency network, then, for implications of these assumptions, see the following in RIL-1101: Table 2, H-culture-12; Table 4, H-ProcState-4; Table 8, H-SR-12-14; Table 9, H-SRE-2G2; Table 14, H-SAE-1G1 item 1, H-SAE-7.1.

C.2 Reference model for hazard analysis in development lifecycle

Hazard analysis of a digital safety system is part of its safety analysis activities, which are independent from the mainstream development activities, within which also some form of HA and V&V occurs. Nevertheless, the independent HA is interrelated with associated systems engineering activities, as depicted in Figure 9 and charted in Table 21. The independent team may engage the initial HA-team in review and walks through its work products.

In the context of hazards contributed through engineering deficiencies, a contributor may be detected and controlled in (a) the mainstream system development, which includes some form of HA [4] and V&V; (b) independent V&V processes; or (c) independent HA. In general, the higher the quality of the upstream processes, the smaller will be the hazard space downstream, and the lower will be the amount of hazards within downstream work products. On the other hand, ill-controlled upstream processes could render downstream V&V and HA infeasible. Recognizing the wide variation in the practice of upstream system engineering, for the purpose of consistent comprehensible concise treatment of the inter-relationship of HA with the other processes, the state-of-the-art in system and safety engineering is used as a baseline and reflected in the lifecycle reference model, depicted in Figure 9. The reference model is derived from [1] for integrity level 4. Thus, the independent HA activities are characterized under the following premises:

1. Mainstream system development activities are performed in accordance with the specifications of their respective processes.
2. Resources used in these development activities are qualified to meet their respective specified requirements or criteria.
3. V&V processes fulfill the objectives stated in Section 1.4 of [1].
4. Verification activities (on the object of verification) confirm that the requirements specified for that object are satisfied.

   4.1. Anomalies are detected as early in the lifecycle as possible, in accordance with [1].

4.2. Detected anomalies are resolved in accordance with [1]

5. Supporting audits of the process activities in execution examine whether these activities are being performed in accordance with their specifications, using resources that conform to their respective requirements. Deficiencies are corrected promptly.
6. Mainstream validation activities confirm that the various specifications collectively satisfy the requirements intended from the NPP level safety analysis.
7. The "object" of analysis has passed its V&V criteria.

Under premises 1-7 stated above, independent HA activities provide an independent search for the remaining "conditions having the potential for functional degradation of safety system performance" (known as hazard identification) and seek their control (e.g., avoidance or elimination) through corresponding requirements and constraints. This search starts from the safety function of concern, first identifying the direct hazards and, then, for each hazard, progressing "upstream" through the dependency paths to identify the contributory hazards. The independent HA perspective is broader than the mainstream activities; for example, it may re-examine:

- Interpretations of a requirement specification;
- Flow-down of derived requirements and constraints;
- Flow-down of quality requirements[116];
- Premised validity of the process specifications and resource qualification criteria;
- Other assumptions.

To the extent that premises 1-7 stated above are not satisfied, the difference results in additional burden on the independent HA activities, requiring commensurate additional skills and effort. Also see Section 2.3.8.

A regulatory review of HA may be viewed as yet another round of independent HA. Thus, the review activities follow the same pattern.

[116] These are also known as "non-functional" requirements.

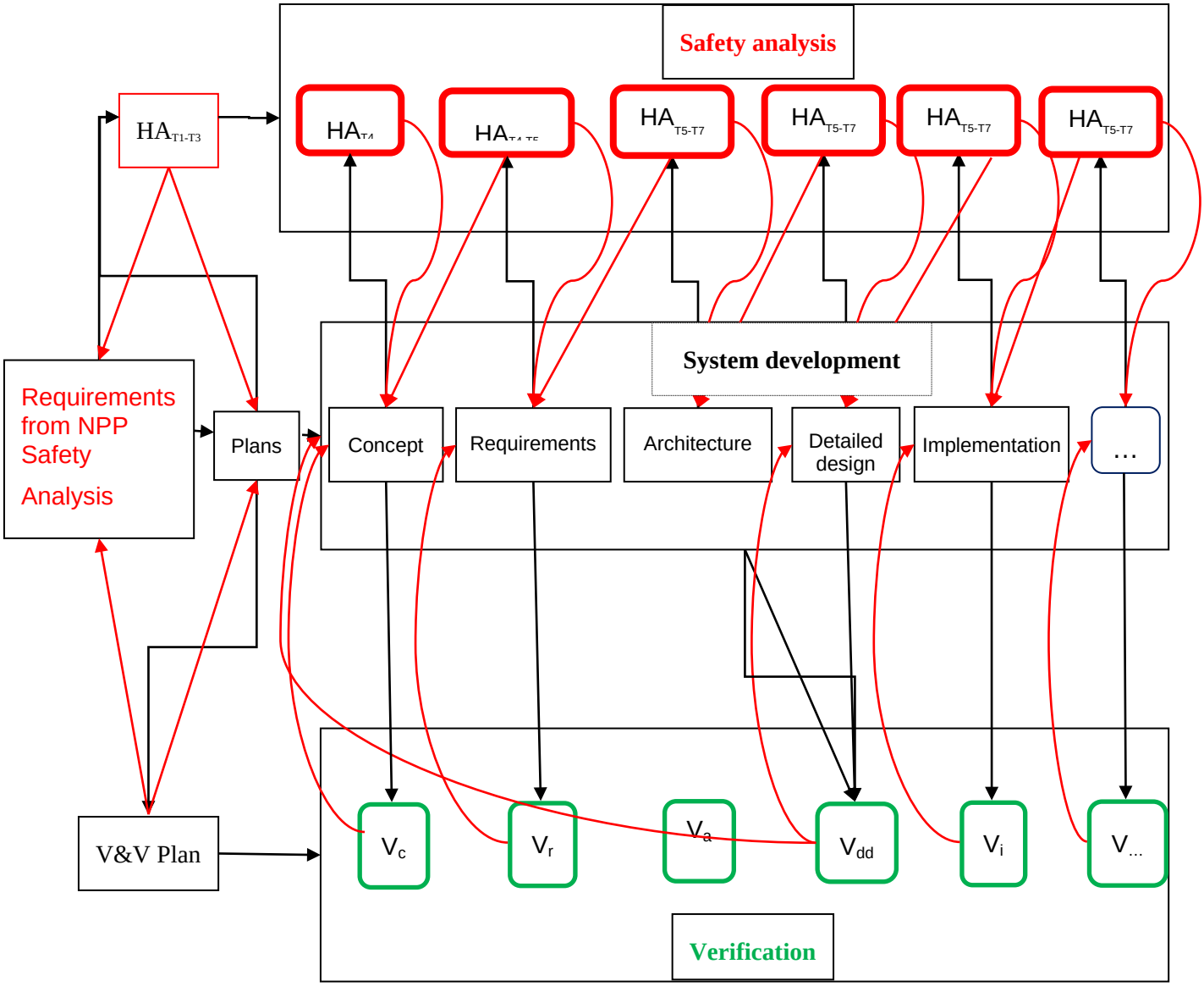


Figure 9: Hazard analysis in relation to development lifecycle and verification activities

C.3 HA tasks – an example set

Referring to Table 21, tasks T1-T3 start in the planning phase of the system engineering lifecycle; however, at every change, the plans are reviewed to identify corresponding changes needed.

Task T4 is started in the concept phase of the system engineering lifecycle. In a "green-field" concept, the information available may only be a functional concept. Yet, it is sufficient to develop the questions to be addressed from the HA perspective, accomplished through the "hazard logging" process. In this case, task T4 may be iterated many times, as the concept evolves. Systematized management of change and configuration (e.g., through minor or internal version identifiers) enables recorded, track-able rationale underlying the evolution path. In a modification of an existing NPP, the concept may be much more developed (e.g., a proposed NPP-level I&C re-architecture), enabling more detailed investigation for the identification of (contributory) hazards.

When the system concept and requirements specification become stable, task T4 transitions into T5, at the start of which, the term "object" refers to the system requirements specification (corresponds to task 203 in [5]). Tasks T5 and T7 are iterated as the system architecture evolves. The iterations include task T6, when a lower level of integration is identified in the system architecture.

Table 21: HA activities and tasks - a reference model

HA activity / task	Input	Output	Remarks. References.
T1. Generate baseline HA plan for all lifecycle phases.	1. Concept [1], incl. interactions with and dependencies on its environment. 2. Requirements from NPP level safety analysis. 3. Premises & assumptions upon which the expected outcome depends, incl. conditions & modes of operation and maintenance. 4. Plan to validate assumptions. 5. Consequences of behavior shortfalls, incl. invalid assumptions/premises. 6. Overall V&V plan, incl. HA. 7. Mainstream development plan. 8. Corresponding information about or from entities in the dependency paths (e.g., up the supply chain).	Baseline[117] HA plan.	Adapted from [1] Table 1a Tasks 7.1:1-4 and Task 101.2.2 in [5].
T2. Identify dependencies of HA plan (e .g. other information; resources; dependencies on supply chain)		Dependencies of plan.	Adapted from: [1] Table 1a Tasks 7.1:1-4; [5].
T3. Evaluate other plans, following the dependencies identified above. T3.1. Coordinate information exchanges with HA activities (e.g., timing; semantic compatibility; format).		1. Evaluation report. 1.1. Deficiencies. 1.2. Changes needed. 1.3. Request for additional information (RAI).	Adapted from [1] Table 1a Tasks 7.1:1-4. 7.4, 7.5.
		2. Rejection or Acceptance (incl. phase-advance clearance)	Adapted from [1] Table 1a Tasks 1-4.
		3. Revision to HA plan as needed.	Adapted from [1] Table 1a Tasks 7.1:1-4.
T4. Understand HA-relevant characteristics of the object to be analyzed; examples: 1. Differences from previously licensed systems. 2. Exposure to unwanted interactions.	Items above + 9. Other requirements allocated to the object. 10. Non-safety related constraints on the object. 11. Relationship	1. Revision to HA plan. 2. Addition to hazard log. [15] 3. Change needed; examples: 3.1. Making	Adapted from [1] Table 1a Tasks 7.2:(1)a, f, g), (2)b,d), (3)a,b) and Tasks 201-202 in [5],

[117] While mainstream HA produces the baseline, independent HA identifies changes needed.

3. Presence of functions not needed for the primary safety function. 4. Division of work and communication challenges across organizational units/interfaces. 5. Compatibility of lifecycle models, processes, information-exchange interfaces, etc. 6. Qualification and compatibility of tools across these interfaces. 7. Compatibility of conditions of use for reused objects. 8. Correct, complete flow-down or decomposition or derivation of requirements. 9. Identification of dependencies (e.g., feedback paths; hidden or obscure couplings). 10. Premises and assumptions – explicit and implicit. 11. Other challenges to analyzability.	with NPP-wide I&C architecture. 12. Distribution of responsibilities across organizational units/interfaces. 13. Provisions for information exchange across organizational units/interfaces. 14. Lifecycle models; processes; resources (e.g., tools; competencies); information exchange interfaces. 15. Identification of reused objects and conditions of use. 16. Explicit record of dependencies. 17. Prior HA results, if any.	assumptions explicit; 3.2. Improvement in knowledge of dependencies. 3.3. Making lifecycles, processes compatible; 3.4. Making information-exchange interfaces compatible; 3.5. Consistency across automation and human roles/ procedures. [7] 3.6. Qualification of reused objects (e.g., tools); 3.7. Change in allocation of a requirement; 3.8. Other constraints; 3.9. Other derived requirements. [13]; 4. RAI	
T5. Analyze object[118] for (contributory) hazards. See corresponding section and table in RIL-1101. For a safety system or its element, it includes, for example, search for: 1. Single point failure; 2. Common mode dependency; 3. Common cause dependency.	Items above + Information specific to object of analysis (see Section C.1.2).	1. Addition to hazard log. 2. Change needed. Examples: 2.1. See in T4; 2.2. Derived requirement (on process) to prove that a contributing hazard cannot occur. 2.3. Derived requirement or constraint on object. 3. Rejection | Acceptance (incl. phase-advance clearance) 4. Revision to HA plan as needed 5. RAI	Adapted from [1] Table 1a Tasks 7.1:5-6; Tables 1b and [1]. [14]
T6. Integrate analyses from lower levels in the integration hierarchy and contribution paths up to the top-level analysis.	Items above + information needed about inter-object dependencies for overall system HA	As in T5.	Adapted from [1] Table 1a Task 7.1:7; [1].

[118] Examples of objects: Work product from any phase in the development lifecycle; Work product for the top-level digital safety system; some element in a lower level of integration; associated processes; associated resources; any other entity in the dependency paths (e.g., in the supply chain).

T7. Analyze change proposal (e.g., hazard control proposal).	Change proposal, including information on which it depends (e.g., items listed above).	As in T5.	Abstracted from [1]

C.3.1 Evaluating the quality of HA output

The quality of the HA output depends upon three major factors:

1. Competence – see Section C.4.
2. Quality of the input(s) – see Section C.5.
3. Technique – see Section C.6.

Evaluation of the HA plan is based on the degree to which the planned HA fulfills the following objectives:

1. Identify all hazards.
   1.1. Identity the constraints on the system and its environment, which would enable item 1.
2. Identify all contributory hazards.
   2.1. Identity the constraints on the system and its environment, which would enable item 2.
3. Identity the constraints needed to control the identified (contributory) hazards.

Consequently, evaluation of a selected HA technique is based on its ability to fulfill the objectives stated above and identifying the associated critical conditions, namely:

1. A specification of the competence required to apply the technique, such that the competence can be evaluated with consistency.
2. A specification of the information required to apply the technique, such that the object of analysis can be evaluated with consistency.

Criteria to evaluate HA ouput[119]

1. Completeness
   1.1. Analysis for all known hazards and contributors, including lessons learned from prior experience.
   1.2. Demonstration of a systematic approach to HA, supported by evidence and reasoning.
2. Demonstrated consistency in the analysis of identified hazards and contributors.
3. Consistency with assumptions used.
4. Reference to inputs used.

C.3.2 Hazard identification and logging

Hazard identification, especially in the concept phase, requires extra-ordinary individual capabilities, teamwork, and a conducive organizational culture (see Appendix F). If any analyst or contributor to HA perceives a safety concern, a hazard, or a contributory hazard, the individual is encouraged to express it. The expressed item is recorded in a “hazard log” without immediate evaluation. Sometimes, a team engages in brainstorming to stimulate thought and encourage expression. The “hazard log” [15] is a means of tracking an item from initial expression to final disposition and closure. An “entry” is never deleted. All the related information may be in a single document or it may be distributed across a set of linked databases; in any case, an analyst is able to make an entry readily.

Examples of related information include the following:

[119] Criteria may be applied to the output in any iteration of any stage of the development lifecycle.

1. Information to identify the logged item:
   1.1. Item identifier;
   1.2. Descriptive title;
   1.3. Originator;
   1.4. Origination date;
   1.5. Description;
   1.6. Perceived consequence/effect of inaction;
2. Information to track progress:
   2.1. Action plan (from origination to closure);
   2.2. Action assignee(s);
   2.3. Status of progress in the action plan (e.g.,
      2.3.1. Identified change needed to eliminate hazard);
   2.4. Basis to allow closure (e.g.
      2.4.1. Evaluation revealed that hazard control is already in place.
      2.4.2. Evaluation resulted in restatement of the hazard (another entry in the hazard log);
      2.4.3. Addition of a constraint or derived requirement in the system engineering activities;
   2.5. Date of closure;
   2.6. Name and Signature authorizing closure.

Every addition or modification of a constraint or (derived) requirement is a configuration controlled item with associated change controls.

When the object is the overall system, the corresponding HA task is the exercise of the selected HA technique (see Section C.6) on the information available about the object (see Section C.5). Execution of this process may assist in the evaluation of some other item in the hazard log; or may raise a new concern, which is then entered in the hazard log.

C.3.3 Evaluation of a logged hazard

Whereas published standards and handbooks (whose scope includes mixed-criticality systems) suggest evaluation in terms of levels of severity and likelihood of occurrence, in the RIL-1101 context, the severity of the loss of a safety function is of the highest level and, for systemic causes, the analysis first seeks their correct identification and then, pursues their elimination or avoidance, as explained next.

In practice, a “quick” filtering or screening evaluation (e.g., see 2.4.1-2.4.2 above) is performed on each logged item, before delving deeper. If an accurate dependency model is available, the evaluation seeks to fit the logged item in the dependency model. The search may reveal that the dependency model is inaccurate (requiring change) or that the logged item is not a (contributory) hazard (leading to its closure). When the logged item is matched to an object in the dependency network (i.e., its sequence in the contributory path is found), a corresponding HA task is formulated and sequenced in accordance with its place in the contributory path.

As the evaluation of a logged item progresses, it may expose inadequacies or uncertainties in the information about the object being analyzed. Figure 10 depicts a structure for reasoning

(adapted from [16]) about these uncertainties[120]. Suppose that the HA team is considering an assertion that the result of their work (e.g., constraint on the object being analyzed) will control the logged (contributory) hazard. Then, the team clarifies its reasoning through discussion, evoking challenges to the assertion and rebuttals to the challenges. The discussion may also reveal inconsistencies in the reasoning. In this manner, the team identifies factors affecting the validity of their assertion. Qualifiers are associated with the assertion; for example:

1. Condition(s) under which the assertion is supported.
   1.1. Uncertainties may be stated as assumptions, for which the truth has to be validated.
   1.2. Changes needed may be stated as constraints to be satisfied.
2. Degree or strength of the assertion: {Strong …. Weak}

The results are recorded, showing how the assertion is supported by the evidence[121], identifying the inference rule to assert the evidence-assertion link, and the technical basis for the rule such as a causal model[122].

[120] Appendix F explains how the process is applied to cross-cultural (e.g., inter-disciplinary; inter-organizational) communication.

[121] It is labeled “grounds” in [16].

[122] It is labeled “backing” in [16].

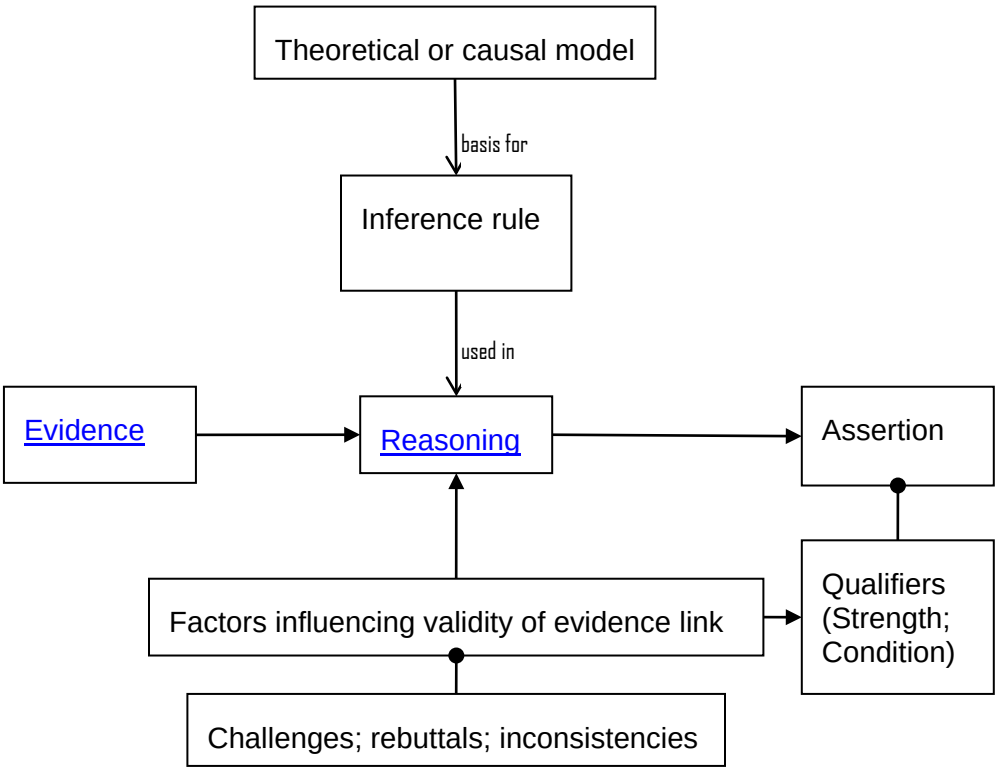


Figure 10: Structure to reason about the contribution to a hazard

If the evaluation results in a conclusion that the logged item is not a hazard, it is recorded, including the information depicted in Figure 10 (e.g., the reasoning, along with unresolved dissenting positions, if any, in the form of conditions). A resolution process ensures that the analysis, evaluation, resolution and disposition of the issue are performed in a timely and effective manner.

C.4 Effect of competence on quality of HA work products

When HA is performed on an early-stage concept, with little explicit information in the concept, the "competence" factor (see Section C.3.1) is most dominant. For example, the analyst has to elicit information about assumptions and dependencies through systematic enquiry, devised for the circumstances. Based on this information, the analyst would have to construct an analyzable model of the dependencies (e.g., control structures, showing feedback paths, interactions, and nested levels). These activities require extremely high competence. For an approach to competence management, see [17], in which reference 7 is a technical competence framework developed through wide consultation in the UK.

Competence is a critical factor - see in RIL-1101 Table 1 items H-0-2G{0, 1, 2}, H-0-3G1;

Table 2 H-culture-6G3, Table 10 H-SRE-1G{1,2,3}. Competence to perform HA of an NPP digital safety system includes a complement of the following (not necessarily in one person):

1. Proven self-learning[123] ability, assimilating needed new knowledge in a scientifically sound framework.
   1.1. Education equivalent to a master's degree level knowledge of safety critical industrial automation systems engineering;
   1.2. Ability to recognize the knowledge needed and limitations of one's knowledge.
   1.3. Ability to fill one's knowledge gaps through self-study, supplemental training, and consultation with experts.
2. Reasoning capability (see Figure 10);
   2.1. Objectivity. (Also see item 9).
   2.2. Ability to abstract and generalize from one context and apply to another.
   2.3. Ability to recognize fallacies in some chain of reasoning.
3. Continuing update of professional knowledge through training; examples:
   3.1. Application domain: How an NPP works (energy conversion from fuel to power on the grid); heat exchange; critical functional elements, processes and process state variables in an NPP and their inter-dependencies; associated (contributory) hazards; study of operating experience (event reports; root cause analysis reports).
   3.2. Industrial automation domain: Elements for sensing, actuation, computation; control logic; communication; software/firmware; power; associated (contributory) hazards; study of operating experience (event reports; root cause analysis reports).
   3.3. Science and engineering of distributed systems, including computation, communication.
   3.4. Hazard and safety analysis and assurance methods and techniques for such systems.
4. Experience in analysis of systems similar in criticality, functionality, and configuration:
   4.1. Good performance under the guidance of an expert in hazard analysis.
   4.2. Good performance independently.
5. Strongly safety conscious. See Appendix F.1 and F.3.
6. Communication skills in group activities (see Appendix F.4) – examples:
   6.1. Ability to communicate effectively, objectively with stakeholders.
      6.1.1. Succinctness.
   6.2. Ability to listen actively for understanding and learning from others.
   6.3. Ability to elicit information needed.
   6.4. Ability to explain one's reasoning (see Figure 10) to others.
   6.5. Ability to express and explain to others insights from deep knowledge.

[123] When the object being analyzed entails some characteristic, which the analyst has not encountered in past experience, as is often the case in digital safety systems, corresponding learning is needed.

6.6. Ability to develop collective communicative competence. See Appendix F.4.3.

7. Other interpersonal skills and characteristics, supportive of teamwork (see Appendix F.4) – examples:
   7.1. Willingness to recognize and accept weakness in own reasoning.
   7.2. Willingness to explain own reasoning (clearly; succinctly) in the face of opposition.
   7.3. Assistive rather than competitive behavior.
   7.4. Ability to evoke minority viewpoints (concerns or reservations).
   7.5. Ability to understand other team members' reference-frames.
   7.6. Ability to assimilate differences, neutralizing biases.
   7.7. Ability to converge[124] towards objectivity (see Figure 10). See "collective mindfulness" in Appendix F.
   7.8. Other constructive group interaction skills.
8. The complement of competence in the HA team includes breadth and depth.
   8.1. Depth: Individuals having mastery over the respective engineering disciplines, technologies, products or components, and processes, involved in each phase of the system development lifecycle (possibly involving phase-wise changes in team-membership) and respective dependencies.
      8.1.1. Knowledge of respective operating experience (what can go wrong).
      8.1.2. Track record of learning from it (how to prevent what went wrong).
   8.2. Breadth[125]: Individuals are able to understand how their respective roles fit into the overall HA, including the associated inter-dependencies.
      8.2.1. Knowledge of the environment[126] of the safety system and its development.
      8.2.2. Experience in analysis of hazard groups such as those identified in RIL-1101.
      8.2.3. Experience in deriving requirements and constraints to avoid or eliminate contributory hazards.
      8.2.4. Experience commensurate to the functionality and configuration of the system.
9. The HA-team has cultural diversity[127] - supportive of safety.

Inadequate replenishment of requisite competence: The DI&C engineering workforce is changing and so is the environment from which the workforce is being replenished. With the decline in the U.S. manufacturing industry, there has been corresponding decline in its industrial automation development base. Education and training concerning software are driven more by consumer products and information technology (IT) industries than by high-consequence automation. "Development of DI&C systems for the highest level of safety" is a very small, niche in the market.

[124] Through ability to articulate premises and qualifications of claims and how those derive from particular contexts.

[125] Provide continuity to the HA-team across lifecycle phases.

[126] Also see Section 3.4.1.

[127] See reference-frames in item 7.5; examples: belief systems, values, thought processes, paradigms, customs, conventions, language..

C.5 Quality of information input to HA at each development phase

Table 22 provides a broad-brush characterization of the quality of the work products (in terms of information richness) available for HA. For each major lifecycle phase work product, Table 22 compares characteristics in common practice with state-of-the-practice (best in class), and state-of-the-art (leading-edge implementations, not yet scaled up).

Table 22: Characterization of information richness in phase work products

Row ID	Work product of lifecycle phase	Common practice	State of the practice (best in class); examples	State of the art; examples
1	Requirements from next higher level of integration, e.g. from NPP-level safety analysis.	Textual narrative. No configuration-controlled vocabulary. "Flat list" organization (i.e., no explicit relationship across requirements is identified).	Restricted natural language with defined vocabulary and structure across elements of a statement. [18]	Use case scenarios [19].
			SpecTRM-RL [20]	Framework for specification & analysis [21].
			Requirements engineering support in Naval Research Labs [22]. Requirements tables as used for Darlington NPP [23][24]. Models to support mechanized reasoning. Examples: SysML [25].	
2	Plans {Safety plan; V&V plan; HA plan}	Low level of detail; relatively late in the lifecycle.	V&V plan [1] Safety plan [26]-[28]	Integrated safety and security plan.
3	Concept	Combination of (a) block diagram without semantics on the symbols and (b) textual narrative	Models to support mechanized reasoning [29]. (See note 1) SysML [25]; AADL [30] Extended EAST-ADL [31]	META [32]
4	Requirements of digital safety system	See row 1	See row 1	See row 1
5	Architecture of digital safety system	See row 3	See row 3	META [32]
6	Requirements for software in digital safety system	See row 1	[29][33][34]	See row 1
7	Architecture for software in digital safety system	See row 3	See row 3. MASCOT [34] AADL [30]	META [32]
8	Detailed design of software	For application logic: Function block diagram [35]. For platform software: Combination of (a) block diagram without semantics on the symbols and (b) textual narrative.	SPARK [36][37]	META [32] Refinement from architectural specifications
9	Implementation of software (code)	For platform software, including communication protocols: C programming language + processor-	Concept of using safe subset of an implementation language: MISRA C [38][39]	Auto-generation from detailed design.

		specific assembler language	Language for programming FPGAs [40]	

Notes:
1. The models should contain enough information to understand dependencies and propagation paths for contributory hazards.

C.6 Hazard Analysis Techniques – useful extractions from survey

The selection and role of HA techniques (the third factor influencing the quality of an HA product mentioned in Section C.3.1) will depend upon the nature of the system to be analyzed and the quality of the information contained in the various intermediate work products, characterized in Section C.5.

Table 23 summarizes some applicable techniques surveyed. As difficulties and limitations were encountered in the earlier techniques (such as those in the first three rows of Table 23), these techniques were extended, adapted and transformed into newer techniques (such as the ones in the last three rows of Table 23); the references for the latter describe some of the difficulties and limitations encountered in using the earlier techniques. The "salient feature(s)" column identifies concepts found useful. However, the adaptations devised to evolve newer techniques require extraordinary ingenuity; utility of the adaptations is very dependent upon the skills of the analysts.

When HA is applied to an early concept phase, it is called preliminary hazard analysis (PHA) [41][42].

For a broad survey of HA techniques, see [7][43][44], and for additional guidance, see [45]-[49]. For a tutorial overview of HA in relation to safety critical system development, see [51]. These references are not included in Table 23, if technique-specific references are listed.

Table 23: Salient features of techniques relevant to NPP digital safety systems

HA technique		Reference(s)	Salient feature(s)
Acronym	**Expanded name**		
HAZOP(S)	Hazard and operability studies	[8]	Concept of using teamwork, aided by HAZOP process expert. Systematizing enquiry through key words. Systematizing understanding effects through understanding the associated deviations.
FTA	Fault Tree Analysis	[52][53][54]	Representation and understanding of fault propagation paths, when the paths are branches of a tree.
DFMEA	Design Failure Mode and Effects Analysis	[55][56][57] [58]	Representation of faulted behavior of a hardware component for understanding its effect, without requiring knowledge of its internals.
FFMEA	Functional Failure Mode and Effects Analysis	[57][69]	Understanding effect of unwanted behavior of a function of the system, without requiring knowledge of its internals. Useful in concept phase.
FuHA	Functional Hazard Analysis	[7]	
FHA	Fault Hazard Analysis	[43][46] [49]	
CCA	Cause Consequence Analysis	[43][49]	Concept of using causality model to understand fault propagation paths.
W/IA	What if analysis	[47][49]	
CCFA	Common Cause Failure Analysis	[43][46] [49]	
HACCP	Hazard Analysis & Critical Control Points	[50]	Concept of focusing on critical process variables that affect the outcome.
SHARD	Software hazard analysis and resolution	[10]	Adaptation of HAZOP to software, through customization of the key words.

FPTN/FPTC	Fault propagation and transformation network/calculus	[59]	Representation and analysis of fault propagation, when the faults are transformed during propagation, and when there are feedback paths, supporting mechanized traversal and reasoning.
DFM	Dynamic Flowgraph Method	[64]-[66]	Behavior modeling of the system in the finite state machine paradigm facilitates or enables: • Mathematical underpinning. • Analysis of its interactions with environment. • Analysis of dynamic behavior across its elements. • Mechanized traversal. • Mechanized reasoning, esp. if directed cyclic graph.
STPA	System-Theoretic Process Approach	[74]-[76]	• Applicable at concept phase (without a finished design). • Applicable to understanding of organization-culture systems.

HAZOP has been adapted to analyze software [8], and this adaptation has been extended to data flow oriented software architecture [10], and, later, extended to systems with feedback and systems in which the initial fault is transformed into other faults as it propagates [59][59]. These concepts and principles have influenced the AADL [30] error annex, supporting analysis of fault propagation. For an indication of promising research to extend AADL for hazard analysis, see [71].

Recently, a technique similar to the adaptations of HAZOP mentioned above, namely STPA, has been demonstrated in NPP applications [74][75][76].

For a comparative experimental study of six techniques, see [74].

If HA is performed on a state-of-the-practice or state-of-the-art work product, such as the ones shown in Table 22, and if all behavior-influencing assumptions and dependencies were already explicit in a system architecture model, the search for (contributory) hazards could be automated [61]-[65], reducing the dependence on extremely high competence. However, model-based approaches introduce their own contributory hazards [63], to analyze which highly specialized competence is needed.

For adaptation of the concepts in [11], [59] and [59] for HA of device interfaces and, then, HA of operating systems, see [66]-[69].

For an adaptation of the concepts in [59]-[60] to address the fault propagation problem for FPGAs, see early experimental work reported in [70].

For an example of showing freedom from exceptions in software implementations (which are contributing hazards), in addition to showing conformance to specifications, see [37].

For an example of analysis for hazards contributed through timing aspects of multi-core computing processor resources, see [72].

Static analysis tools, such as [37] identify data, information and control flow dependencies in software.

For emerging guidance on HA of complex hardware, such as FPGAs, see [71]. For ongoing developments in this field, track [72].

C.7. References

[1] IEEE Standard 1012-2012, “IEEE standard for system and software verification and validation,” March 29, 2012.

[2] NUREG/CR-7007, “Diversity strategies for nuclear power plant instrumentation and control systems” 2010.

[3] IEEE Standard 603-2009, “IEEE standard criteria for safety systems for nuclear power generating stations” 2009.

[4] Joint Software System Safety Committee, “Software System Safety Handbook – A Technical & Management Team Approach,” December 1999. URL: http://www.system-safety.org/Documents/Software_System_Safety_Handbook.pdf

[5] MIL-STD-882E, “Standard Practice for System Safety,” U.S. Department of Defense, May 11, 2012.

[6] European Committee for Electrotechnical Standardisation (CENELEC), EN 50126, Part 1: 1999, Railway applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS). (A new version is soon to be released; I assume that it will not have changed much in concept.)

[7] Society of Automotive Engineers (SAE), ARP-4761 – Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, 1996. (This standard contains a description of Functional Hazard Analysis (FHA) and of Preliminary System Safety Assessment (PSSA).)

[8] CISHEC (The Chemical Industry Safety and Health Council of the Chemical Industries Association Ltd.), A Guide to Hazard and Operability Studies, 1977.

[9] McDermid, J.A., Nicholson, M., Pumfrey, D.J. & Fenelon, P., (1995), Experience with the application of HAZOP to computer-based systems, COMPASS '95: Proceedings of the Tenth Annual Conference on Computer Assurance, Gaithersburg, MD, pp. 37-48, IEEE, ISBN 0-7803-2680-2.

[10] Redmill F., Chudleigh, M., Catmur J., System Safety: HAZOP and Software HAZOP. John Wiley and Sons Ltd., Chichester, U.K., 1999.

[11] McDermid, J.A. & Pumfrey, D.J., (1998), Safety Analysis of Hardware / Software Interactions in Complex Systems, Proceedings of the 16th International System Safety Conference, Seattle, WA, pp. 232-241, System Safety Society.

[12] IEC Standard 61882, “Hazard and Operability Studies (HAZOP Studies) – Application Guide,” International Electrotechnical Commission, First Edition, 2001.

[13] McDermid J.A., Safety critical software, in: Encyclopedia of Aerospace Engineering, Online, Wiley 2012 (accessible via DOI: 10.1002/9780470686652.eae506).

[14] Hawkins R.D., Habli I., Kelly T.P., The Principles of Software Safety Assurance, in Proceedings of the 31st International System Safety Conference, Boston, MA, International System Safety Society.

[15] SMP 11, MoD Hazard Log Requirements, see: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/27584/SMP11v22final.pdf (last accessed 31st July 2013)

[16] Toulmin, Stephen. The Uses of Argument. Cambridge: University Press, 1958

[17] HSE statement on competency management, see: http://www.hse.gov.uk/consult/condocs/competence.pdf (last accessed 4th August 2013)

[18] Hinchey, A,G, et al, “Towards an automated development methodology for dependable systems with application to sensor networks” Performance, Computing, and Communications Conference, 2005. IPCCC 2005. 24th IEEE International, 2005

[19] Allenby, K., Kelly, T., “Deriving Safety Requirements Using Scenarios,” Proceedings of the Fifth International Symposium on Requirements Engineering, p.p.. 228-235, Toronto, Ont, Canada, August 7, 2002.

[20] SpecTRM-RL http://www.safeware-eng.com/software%20safety%20products/features.htm

[21] Day, N.A., Joyce, J.A., “A framework for multi-notation requirements specification and analysis” Proceedings, ICRE 2000. URL http://ieeexplore.ieee.org/ielx5/6907/18574/00855551.pdf?tp=&arnumber=855551&isnumber=18574

[22] Heitmeyer, et al, “The SCR method for formally specifying, verifying, and validating requirements: tool support” ICSE 1997. URL: http://ieeexplore.ieee.org/ielx3/4837/13372/00610430.pdf?tp=&arnumber=610430&isnumber=13372

[23] Parnas D., Madey J., Functional Documents for Computer Programs. Science of Computer Programming, Vol. 25, No. 1, 1995.

[24] Galloway, A., Iwu, F., McDermid, J. A., Toyn, I., On the Formal Development of Safety Critical Software, In: Verified Software: Theories, Tools, Experiments, First IFIP TC 2/WG 2.3 Conference, VSTTE 2005, Zurich, Switzerland, October 10-13, 2005, Meyer, B., Woodcock, J. C. P. (eds.) pp 362-373.

[25] SysML, see: http://www.omgsysml.org/ (last accessed August 1st 2013).

[26] ISO - International Organization for Standardization, BS ISO 26262-2: 2011, Road Vehicles – functional safety, Part 2: Management of functional safety.

[27] ISO - International Organization for Standardization, BS ISO 26262-3: 2011, Road Vehicles – functional safety, Part 3: Concept phase.

[28] ISO - International Organization for Standardization, BS ISO 26262-4: 2011, Road Vehicles – functional safety, Part 4: Product development at the system level.

[29] Despotou G., Alexander R., Kelly T.P., Addressing Challenges of Hazard Analysis in Systems of Systems, 2009, In proceedings of the 3rd Annual IEEE International Systems Conference (SysConf '09), Vancouver Canada, 23-26 March 2009.

[30] AADL, see; http://www.aadl.info/aadl/currentsite/ (last accessed August 1st 2013)

[31] Mader, R., Grießnig, G., Leitner, A., Kreiner, C., Bourrouilh, Q., Armengaud, E., Steger, C., Weiβ, R., “ A Computer-Aided Approach to Preliminary Hazard Analysis for Embedded Systems,” 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, 2011.

[32] OpenMETA tool suite. URL: http://www.army-technology.com/news/newsvanderbilt-university-support-meta-tools-maturation-darpa-avm-programme

[33] Miller, S.P., Tribble, A.,C., Extending the Four Variable Model to Bridge the System-Software Gap, in Proc. 20th Digital Avionics System Conference, DSAC01, Daytona Beach Florida, October 2001.

[34] Simpson H.R., The MASCOT method. Software Engineering Journal, 1(3):103–120, March 1986.

[35] International Electrotechnical Commission, “Programmable controllers – Part 3: Programming languages” IEC 61131-3, ed3.0, 2013.

[36] Barnes J.G.P., High Integrity Software: The SPARK Approach to Safety and Security, Addison Wesley, 2003.

[37] SPARK Pro toolset, see: https://www.adacore.com/sparkpro/ (last accessed August 2nd 2013)

[38] MISRA C, see: http://www.misra.org.uk/MISRAHome/MISRAC2012/tabid/196/Default.aspx (last accessed August 1st 2013)

[39] LDRA MISRA C toolset, see: http://www.ldra.com/en/solutions/by-standard-adherence/misra (last accessed August 2nd 2013)

[40] Conmy P.M., Pygott C., Bate I.J., VHDL Guidance for Safe and Certifiable FPGA Design, IET System Safety Conference, October 2010.

[41] Gowen, L.D., Collofello, J.S., Calliss, F.W., “Preliminary Hazard Analysis for Safety-Critical Software Sysems,” Proceedings from IPCCC’92, 1992.

[42] Safeware Engineering Corporation, “Preliminary Hazard Analysis,” < http://www.safeware-eng.com/Safety%20White%20Papers/Preliminary%20Hazard%20Analysis.htm>, December 6, 2011.

[43] Ericson II., C.A., “Hazard Analysis Techniques for System Safety,” John Wiley and Sons, August 24, 2005.

[44] U.S. Nuclear Regulatory Commission, “Software Safety Hazard Analysis,” NUREG/CR-6430, Washington, DC, February 1996 (Agencywide Documents Access and Management System (ADAMS) Public Legacy Library Accession No. 9602290270).

[45] National Aeronautics and Space Administration, “NASA Software Safety Guidebook”, NASA-GB-8719.13, Washington, DC, March 31, 2004.

[46] U.S. Air Force, “The Air Force System Safety Handbook”, Kirtland AFB, NM, July 2000.

[47] European Strategic Safety Initiative, “Guidance on Hazard Identification,” European Strategic Safety Initiative – Safety Management System and Safety Culture Working Group, March 2009.

[48] Ippolito, L., Wallace, D., “A Study on Hazard Analysis in High Integrity Software Standards and Guidelines,” National Institute of Standards and Technology, NISTIR 5589, Gaithersburg, MD, January 1995.

[49] System Safety Society, “System Safety Society Handbook: A source Book for Safety Practitioners,” The System Safety Society, 1993.

[50] Hazard analysis and critical control points HACCP principles and application guidelines, URL: http://www.fda.gov/Food/GuidanceRegulation/HACCP/ucm2006801.htm

[51] Johnson, Chris, “Safety Critical System Development”, University of Glasgow – Department of Computing Science, Part II of Notes, October 2006.

[52] U.S. Nuclear Regulatory Commission, Fault tree handbook (NUREG 492). URL: http://www.nrc.gov/reading-rm/doc-collections/nuregs/staff/sr0492/sr0492.pdf

[53] NASA, “Fault tree handbook with aerospace applications.” URL: http://www.hq.nasa.gov/office/codeq/doctree/fthb.pdf

[54] Park, G.Y., Koh, K.Y., Jee, E., Seong, P.H., Kwon, K.C., and Lee, D.H., “Fault Tree Analysis of KNICS RPS Software,” Nuclear Engineering Technology, Vol. 41, No. 4, May 2009.

[55] SAE J1739, “Potential Failure Mode and Effects Analysis in Design (Design FMEA), Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes (Process FMEA), 2009” URL: http://standards.sae.org/j1739_200901/

[56] NASA, “Standard for Performing a Failure Mode and Effects Analysis (FMEA) and Establishing a Critical Items List (CIL) (DRAFT): Flight Assurance Procedure (FAP)- 322-209,” Nov. 2011, Available: rsdo.gsfc.nasa.gov/documents/Rapid-III-Documents/MAR-Reference/GSFC-FAP-322-208-FMEA-Draft.pdf

[57] P.L. Goddard, “Software FMEA Techniques,” Proceedings of the Annual Reliability and Maintainability Symposium, IEEE, 2000, pp. 118–123 Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=816294.

[58] G.-Y. Park, “Software FMEA Analysis for Safety Software,” International Conference on Nuclear Engineering, Brussels, Belgium: ASME

[59] Fenelon P., McDermid J.A., Nicholson M., Pumfrey D.J., Towards Integrated Safety Analysis and Design, ACM Applied Computing Review, Aug. 1994.

[60] Wallace M., Modular Architectural Representation and Analysis of Fault Propagation and Transformation, Electronic Notes in Theoretical Computer Science 141 (2005) 53–71.

[61] Hecht, H. and Menes, R., “Software FMEA automated and as a design tool” SAE 08WATC-0023, 2008.

[62] Hecht, H., An, X., Hecht, M., “Computer aided software FMEA for unified modeling language based software.”

[63] UML Safety Analysis, see: https://www.ibm.com/developerworks/community/blogs/BruceDouglass/entry/safety_analysis_with_the_uml8?lang=en (last accessed August 2nd 2013)

[64] Garrett, C. and Apostolakis, G., “Context in the risk assessment of digital systems” Risk Analysis Vol. 19 No. 1 1999.

[65] Garrett, C. and Apostolakis, G., “Automated hazard analysis of digital control systems” Reliability Engineering and Safety Society 77 (2002) 1-17

[66] Aldemir, Guarro, et al, “A Benchmark Implementation of Two Dynamic Methodologies for the Reliability Modeling of Digital Instrumentation and Control Systems,” NUREG/CR-6985, U.S. Nuclear Regulatory Commission, Washington, D.C. (2009).

[67] McDermid, J.A. & Pumfrey, D.J., (2000), Assessing the Safety of Integrity Level Partitioning in Software, Lessons in System Safety: Proceedings of the Eighth Safety-critical Systems Symposium, Southampton, UK, Ed. Redmill, F. & Anderson, T., pp. 134-152, Springer, ISBN 1-85233-249-2.

[68] Conmy P.M., Crook-Dawkins S.K., A Systematic Framework for the Assessment of Operating Systems, Safety-Critical Systems Symposium, Warwick, UK, February 2004.

[69] Conmy P.M., Safety Analysis of Computer Resource Management Software, PhD Thesis, University of York, 2005.

[70] Conmy P.M., Bate I.J., Component-Based Safety Analysis of FPGAs, IEEE Transactions on Industrial Informatics, Vol 6, No 2, May 2010. pp 195-205

[71] Bozzano M., Cimatti A., Katoen J-P., Nguyen V.Y., Noll T., Roveri M., Safety, Dependability and Performance Analysis of Extended AADL Models, The Computer Journal, Vol. 54 No. 5, 2011

[72] ParMERASA, see: http://www.parmerasa.eu/ (last accessed August 2nd 2013)

[73] iScade, see: http://iscade.co.uk/ (last accessed 31st July 2013)

[74] Torok, R. and Geddes, B. "Systems Theoretic Process Analysis (STPA) Applied to a Nuclear Power Plant," MIT STAMP Workshop, March 26-28, 2013. <http://psas.scripts.mit.edu/home/wp-content/uploads/2013/04/02_EPRI_MIT_STAMP_Mar2013.pdf>

[75] Song, Yao, "Applying system-theoretic accident model and processes (STAMP) to hazard analysis" McMaster University dissertation. URL: http://digitalcommons.mcmaster.ca/opendissertations/6801/

[76] Thomas, J. et al, "Evaluating the safety of digital instrumentation and control systems in nuclear power plants" November, 2012. URL: http://sunnyday.mit.edu/papers/MIT-Research-Report-NRC-7-28.pdf

C.8. Bibliography[128]

[77] [129]Atchison B., The Integration of Safety Analysis and Functional Verification Techniques for Software Safety Arguments, 2004, PhD Thesis, University of Queensland.

[78] Chambers, L., "A Hazard Analysis of Human Factors in Safety-Critical Systems Engineering," 10th Annual Workshop on Safety-Related Programmable Systems (SCS-05), Conference in Research and Practice in Information, Vol. 55, Sydney, Australia, 2005.

[79] Alexander, R., Kelly, T., "Can We Remove the Human from Hazard Analysis," University of York.

[80] ISO/IEC 15026-2:2011, "System and Software Engineering - System and Software Assurance - Part 2: Assurance Case.

[128] References Reviewed but not Yet Cited in Appendix C

[129] c

Appendix D: REFINEMENT

Enabling verifiability earlier in the lifecycle through stepwise refinement

Author: Professor Dr. Manfred Broy, Technische Universität München
http://www4.in.tum.de/~broy/

Integrative editing by Sushil Birla, Senior Technical Advisor for I&C, U.S. Nuclea Regulatory Commission

D.1 Purpose and Scope

This appendix explains refinement (see section 2) as an enabler for the verifiability and thus, assurability of a system; this is the usage context in RIL-1101 (see Table 7 item # H-S-1.1G1.4).

The scope of this appendix is limited to the introduction of the kind of refinement needed to support the purpose stated above (rather than covering refinement of all kinds found in literature). For example, excluded from the scope is the case where a specification is expressed through an informal language and informal diagrams. Such a specification may be ambiguous and its meaning may differ, depending on individual subjective judgment, as illustrated in the following situation:

> When a system[130] is conceived, typically its specification is expressed in a language natural to the conceiver (i.e., informal language). The specification may be incomplete (i.e., not all the properties of the system are expressed, basing the economy of expression on an implicit context), inconsistent, and ambiguous. Different individuals with different mental models (e.g., of the conceiver's implicit context and assumptions) might have different interpretations, using their different mental models and judgment to fill in the implicit or missing information in different ways. Transforming the informal description into a complete, consistent, unambiguous[131], correct set of requirements specification may require engineering activities (e.g., elicitation; system-level hazard analysis; validation) other than refinement [8].

This restricted usage of refinement reduces sources of uncertainty in the verification process. This benefit is further discussed in Section 3 and the commensurate required restrictions are introduced in Section 4.

D.2 Abstraction and refinement

Abstraction is a view of an object that focuses on the information relevant to a particular purpose and ignores the remainder of the information [2].

Conversely, refinement is a detailed description that conforms to another (its abstraction) … perhaps in a somewhat different form … [3].

Two specifications S0 and S1 are in a refinement relation if everything described by S0 can also be concluded by specification S1. This relation also ensures that S1 does not add any behavior not included in S0 (i.e., no additional behavior is visible at the external interface).

Stepwise refinement decomposes the development process into a sequence of transformation steps, as depicted in Figure 11, where each successive step refines its input specification [4],

[130] "System" refers to the final product (i.e., the implementation installed in a plant).

[131] Typically, a formal language is used to eliminate ambiguity and facilitate mechanized reasoning.

[5]. Each transformation step entails some design decisions [6]. In other words, it reduces the design space available for the remaining steps.

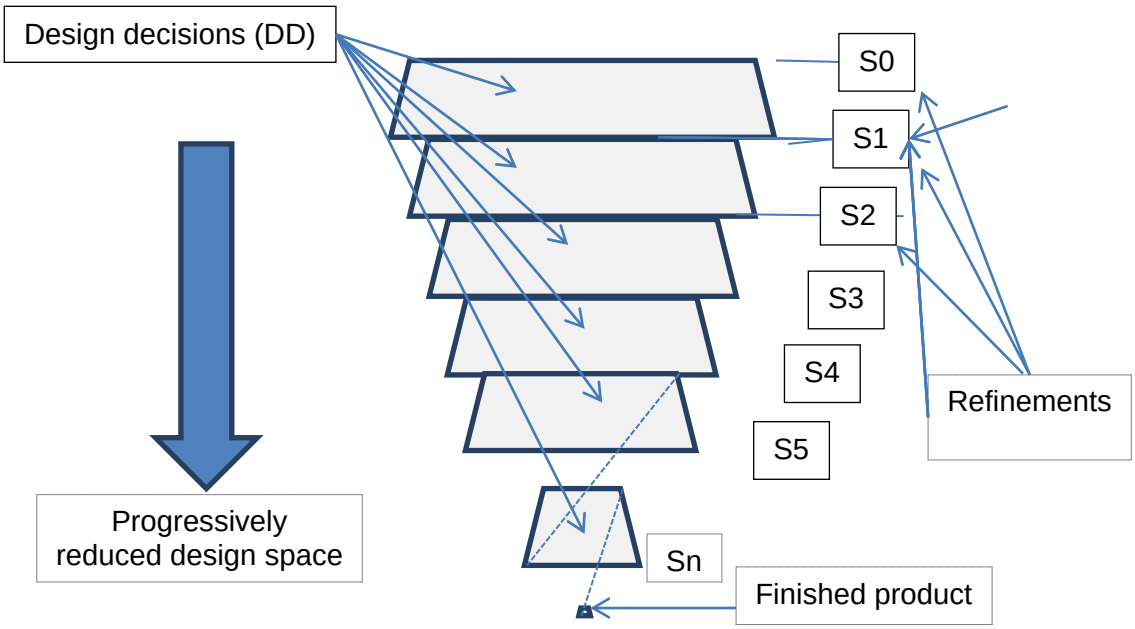


Figure 11: Stepwise refinement: design decisions are made in small steps

The concept of Refinement, in its broadest sense, is applied to the specification of many aspects of a system and many kinds of its elements, such as:

- Data element (see [7])
- Data structure
- Function
- Requirement
- System interface and interface behavior
- System architecture
- Hardware element
- Software element
- Human element
- System implementation
- Process
- Procedure (e.g., operating instructions)

Some simple examples of refinement are given in Table 24.

Table 24: Simple examples of refinement

Type of information	Example of abstract level	Example of refined level
Data	Length.	Length in SI units; value has a specified precision level.
Data structure	Sequence of a given length	Bounded one-dimensional array
Structured data	Sequence of last 10 measured values of distance (length) in SI units.	One-dimensional array of length 10, where each element can be stored (written) or retrieved (read) as a value of length in SI units, but internally the data is stored in a compact form.

Structured data	Location of a point, A, in space, with respect to a given origin and some reference frame.	Location and orientation of point A with respect to a Cartesian reference coordinate frame, C0; all measurements are in SI units : location is designated A_{C0}
Function	Calculate location of A with respect to another Cartesian coordinate frame C1, using IEEE 754 standard for floating point arithmetic: result designated A_{CI}.	Calculate location of A_{CI} using matrix representation and matrix functions, conforming to IEEE 754: $[A_{CI}] = [A_{C0}] - [C1_{CO}]$

D.3 Motivation for refinement as a constraint on system development

Refinement has supported powerful reasoning in software development; success in its use for program construction leads to its usage in the development of safety-critical software-dependent systems [8]. Refinement (in the restricted sense stated in Section 4) enables "verification by construction" that the original specification and initial constraints are satisfied [1].

This approach supports the concept that system properties can be verified analytically by abstracting the essential information and leaving out all details about the system that are not needed (because these details may render the analysis infeasible). The abstraction has to suit the analytical purpose.

The enabling idea is that in the transformation step from the abstract to the refined specification, the verification performed on the abstract level remains valid also for the refined specification.

This idea can be applied to a sequence of refinement steps: Verification of properties successfully applied to abstractions hold also for their refinements.

In the ideal state (enabling verification by construction), the final product would not have to be tested against the initial specification. Key constraints required in developing a system to enable this ideal are introduced next. To the extent that the ideal is not achieved through the refinement-based analytical verification approach, residual uncertainties would require complementary means of verification.

Stepwise refinement serves as a process to make a sequence of design decisions so as to rule out unsafe choices or choices for which safety cannot be assured (e.g.: the technological base does not exist; the organization does not have the capability). In other words, the design space is progressively reduced in a manner that progressively reduces the hazard space also.

D.4 Mathematical underpinnings

Refinement supports correctness notions in a rigorous way, when used with mathematical underpinnings through refinement calculi. Refinement calculi exist for practically all kinds of formalisms and programming notations in computer science and for a large number of system models.

In a refinement calculus for refinement steps, a "chunk" of design activity is decomposed into elementary steps, such that the specification for the "chunk" is preserved [8].

Refinement calculi introduce a formal refinement relation on the set of specifications as well as rules to deduce and prove refinement types forming a formal calculus. Moreover, refinement calculi often define a number of transformation rules for system specifications that are applied to produce refinements and that guarantee correct refinement steps.

D.4.1 Refinement as logical implication

Logically, refinement corresponds to *implication* – the refined specification satisfies its original specification.

If a refinement specification S0 is refined to specification S1, it connotes that specification S1 expresses more detailed information than specification S0; the logical property formulated by specification S1 *implies* the logical property formulated by specification S0.

Formal specifications are logical predicates on systems and thus we can use the concept of logical implication "⇐" to express a refinement relation:

$$S0 \Leftarrow S1$$

Note that the arrow goes from S1 (the refinement) to S0 (the abstraction), expressing that each property expressed by S0 is *implied* by the property expressed by S1.

The transformation from S0 to S1 is called a *refinement* step. Specification S1 is called a *refinement* of specification S0. Specification S0 and specification S1 are said be in the *refinement* relation.

D.4.2 Useful properties of the refinement relation

Refinement relation is a partial order on the semantics of specifications.

The refinement relation is transitive, reflexive, and antisymmetric – it defines a partial ordering on the (semantics of) specifications of systems and their elements.

The transitivity property is illustrated as follows:

If specification S1 is a refinement of specification S0, i.e.:

$S0 \Leftarrow S1$

and S2 is a refinement of S1, i.e.:

$S1 \Leftarrow S2$

then we conclude that S2 is a refinement of S0.

$S0 \Leftarrow S2$

D.4.3 Sequence of Refinement Steps

In developing a system through the stepwise refinement technique, simple steps of refinement are put together into larger steps. To explain and comprehend the correctness of refinement steps of the form

$$S \Leftarrow S'$$

the differences between specifications in adjacent steps must not be too large and incomprehensible. For example, if

$$S \Leftarrow S'$$

Is a large step,

then it is better to decompose it into a sequence of smaller intermediate steps:

$S \Leftarrow S_1$

$S_1 \Leftarrow S_2$

…

$S_{k-1} \Leftarrow S_k$

$S_k \Leftarrow S'$

These smaller steps guarantee that the larger step

$S \Leftarrow S'$

is a correct refinement step based on the fact that the refinement relation is transitive.

D.4.4 Refinement and Decomposition

In a design step a "hard-to-analyze" system, represented with its model M, is decomposed into a number of "easier-to-analyze" (model) elements M_1, M_2, …, M_k.

D.4.4.1 Composing and Decomposing Interfaces

Composition is an operation on syntactically compatible system interfaces; let $[I \blacktriangleright O]$ denote the set of interface behaviors; composition is defined by the operator

$$\otimes : [I1 \blacktriangleright O1] \times [I2 \blacktriangleright O2] \rightarrow [I \blacktriangleright O]$$

The operator $\otimes$ induces a composition operation on specifications [10].

To express this step of decomposition formally we use the composition operator $\otimes$ for systems such that

$$M = M_1 \otimes M_2 \otimes \ldots \otimes M_k$$

This equation expresses both that M is the result of composing M_1, M_2, M_k and that M may be correctly decomposed into the elements M_1, M_2, … M_k.

Following this scheme a specification S is decomposed into a number of specifications S_1, S_2, S_k of its system elements. Generalizing the composition to specifications we write

$$S_1 \otimes S_2 \otimes \ldots \otimes S_k$$

for the specification of all the systems $M_1 \otimes M_2 \otimes \ldots \otimes M_k$ where the elements M_1, M_2, M_k fulfill the specifications S_1, S_2, S_k respectively.

Such a step of decomposition of a specification into specification of system elements is called a refinement step if

$$S \Leftarrow S_1 \otimes S_2 \otimes \ldots \otimes S_k$$

holds.

D.4.4.2 Compositionality of Refinement

Compositionality of refinement guarantees for systems composed of a set of elements that refinements of the specifications of system elements guarantee refined system specifications [1][11][12][13]. A refinement relation is called **compositional** for a given concept of composition, if specifications of systems given by a composition of specifications of their elements are in the refinement relation to systems that are given by a composition of refinements of the specifications of the elements [13]. In the literature compositionality of refinement is sometimes also called modularity of refinement.

If we replace in a larger system an element that is required to fulfill specification A (and if for the correctness of the system this is all that is required) then replacing the element by one fulfilling

specification B is correct and maintains the correctness, if such an element fulfilling specification B also fulfills specification A. Formally, given a specification S, a decomposition $S_1 \otimes S_2 \otimes \ldots \otimes S_k$ which is a refinement

$$S \Leftarrow S_1 \otimes S_2 \otimes \ldots \otimes S_k$$

and refinements R_1, R_2 … R_k of the specification S_1, S_2 … S_k; if the refinement relation is compositional for composition we then can conclude:

$$S_1 \otimes S_2 \otimes \ldots \otimes S_k \Leftarrow R_1 \otimes R_2 \otimes \ldots \otimes R_k$$

and by transitivity of refinement

$$S \Leftarrow R_1 \otimes R_2 \otimes \ldots \otimes R_k$$

Compositional refinement also captures the idea of **compatibility** (replaceability) of a system or its elements. Consider system designs given by a composition of elements where the design is correct as long as the elements are correct in terms of given specifications. Compositional refinement guarantees that the replacement of a specification of an element by its refinement yields a refined design.

D.4.4.3 Example

Figure 12 depicts an example of architectural refinement. The top-level system is represented through its model M and its behavior, through its specification S. The system model is decomposed into modeling elements M1, M2, and M3 and their respective behaviors, through S_1, S_2, and S_3. Their combined behavior results in the behavior S, and does not produce any behavior not specified in S.

$$S \Leftarrow S_1 \otimes S_2 \otimes S_3$$

Note that the refined system contains more information – in this case about the architectural design decomposing model M into three modeling elements M1, M2, and M3 specified by S_1, S_2, and S_3.

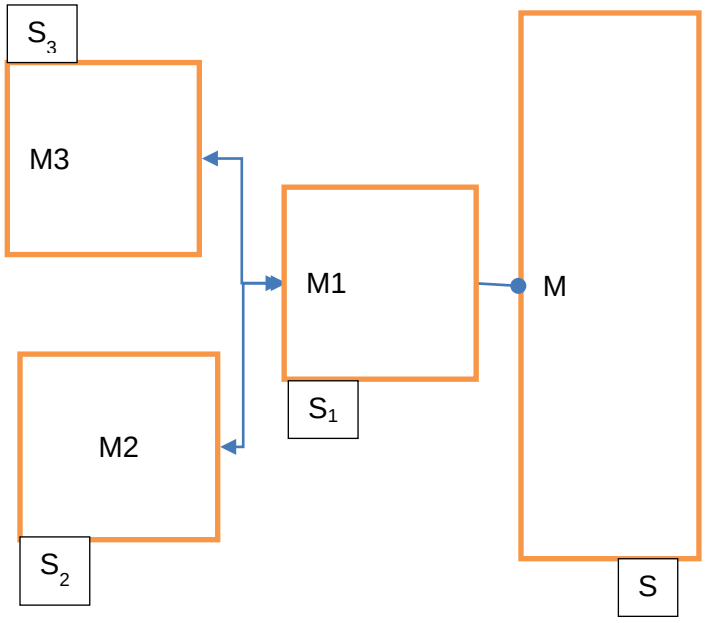


Figure 12: Example of architectural refinement

D.5 References for Appendix D

[1] J. Woodcock, J. Davies: Using Z: specification, refinement, and proof. Prentice-Hal, NJ, USA 1996.

[2] IEEE Standard Glossary of Software Engineering Terminology 610.12-1990. In IEEE Standards Software Engineering, 1999 Edition, Volume One: Customer and Terminology Standards. IEEE Press, 1999.

[3] Desmond F. D'Souza, Alan Cameron Wills. *Objects, Components and Frameworks with UML: The Catalysis Approach*. ISBN 0-201-31012-0, Addison-Wesley, 1999.

[4] M. J. Butler: Stepwise refinement of communicating systems. Science of Computer Programming. Volume 27, Issue 2, September 1996, Pages 139–173

[5] J.W. de Bakker, W.P. de Roever, G. Rozenberg (Eds.): Stepwise Refinement of Distributed Systems, Lecture Notes in Computer Science (2nd edition), Vol. 430, Springer, Berlin (1990)

[6] N. Wirth: Program development by stepwise refinement. Communications of the ACM, Volume 14 Issue 4, April 1971, Pages 221-227

[7] J.M. Morris: Laws of data refinement. Acta Informatica, 26 (1989), pp. 287–308

[8] Michael Jackson; *Refinement, Problems and Structures (extended abstract)*; in Proceedings of Dagstuhl Seminar 09381, 13-18 September 2009 – see http://mcs.open.ac.uk/mj665/papers.html

[9] Carroll Morgan, "Programming from specifications" (Prentice Hall, 2nd edition, 1994, ISBN 0-13-123274-6).

[10] M. Broy: A Theory for Requirements Specification and Architecture Design of Multi-Functional Software Systems. Series on Component-Based Software Development – Vol. 2. Mathematical Frameworks for Component Software. Models for Analysis and Synthesis, 2006, S. 119–154

[11] M. Broy: Compositional refinement of interactive systems. Journal of the ACM (JACM) Volume 44 Issue 6, Nov. 1997

[12] M. Broy, K. Stølen: Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement. Springer 2001

[13] M. Broy: A Logical Basis for Component-Oriented Software and Systems Engineering. The Computer Journal: Vol. 53, No. 10, 2010, S. 1758-1782

Appendix E: Checklists to assist hazard recognition

This Appendix is a collection of checklists assimilated from a variety of sources such as [82][83][84]. It is not an exhaustive coverage of hazard sources, categories or groupings, relevant to an NPP digital safety system. The intent is to stimulate thought from different perspectives, leading to recognition of a hazard or a contributor to it.

E.1 Categories of hazard origination

Table 25 is adapted from NASA Reference Publication 1358 [82] Appendix D, organized by categories of hazard origination or source. For each category, Table 25 identifies a variety of effects which may lead to loss.

Table 25: Some categories of hazard origination

Category of hazard origination	Effect which may lead to loss
Acceleration/Deceleration/Gravity	Inadvertent motion Loose object translation Impacts Failing objects Fragments/missiles Sloshing liquids Slip/trip Falls
Chemical/Water Contamination	System-cross connection Leaks/spills Vessel/pipe/conduit rupture Backflow/siphon effect
Common Causes	Utility outages Moisture/humidity Temperature extreme Seismic disturbance/impact Vibration Flooding Dust/dirt Faulty calibration Fire Single-operator coupling Location Radiation Wear-out Maintenance error Vermin/varmints/mud daubers
Contingencies (Emergency Response by System/Operators to "Unusual" Events)	"Hard" shutdown/failures Freezing Fire Windstorm Hailstorm Utility outrages Flooding Earthquake Snow/ice load
Control Systems	Power outage Interfaces (EMI/RFI) Moisture Sneak circuit Sneak software

Category of hazard origination	Effect which may lead to loss
	Lighting strike Grounding failure Inadvertent activation
Electrical	Shock Burns Overheating Ignition of combustibles Inadvertent activation Power outage Distribution back feed Unsafe failure to operate Explosion/electrical (electrostatic) Explosion/electrical (arc)
Mechanical	Sharp edges/points Rotating equipment Reciprocating equipment Pinch points Lifting weights Stability/topping potential Ejected parts/fragments Crushing surfaces
Pneumatic/Hydraulic Pressure	Over-pressurization Pipe/vessel/duct rupture Implosion Mislocated relief valve Dynamic pressure loading Relief pressure improperly set Backflow Crossflow Hydraulic ram Inadvertent release Miscalibrated relief device Blown objects Pipe/hose whip Blast
Temperature Extremes	Heat source/sink Hot/cold surface burns Pressure evaluation Confined gas/liquid Elevated flammability Elevated volatility Elevated reactivity Freezing Humidity/moisture Reduced reliability Altered structural properties (e.g., embrittlement)
Radiation (Ionizing)	Alpha Beta Neutron Gamma X-Ray
Radiation (Non-Ionizing)	Laser Infrared Microwave Ultraviolet
Fire/Flammability—Presence of	Fuel Ignition Source Oxidizer Propellant
Explosive (Initiators)	Heat

Category of hazard origination	Effect which may lead to loss
	Friction Impact/shock Vibration Electrostatic discharge Chemical contamination Lightning Welding (stray current/sparks)
Explosives (Effects)	Mass fire Blast overpressure Thrown fragments Seismic ground wave Meteorological reinforcement
Explosive (Sensitizes)	Heat/cold Vibration Impact/shock Low humidity Chemical contamination
Explosives (Conditions)	Explosive propellant present Explosive gas present Explosive liquid present Explosive vapor present Explosive dust present
Leaks/Spills (Material Conditions)	Liquid/cryogens Gases/vapors Dusts—irritating Radiation sources Flammable Toxic Reactive Corrosive Slippery Odorous Pathogenic Asphyxiating Flooding Runoff Vapor propagation
Physiological (See Ergonomic)	Temperature extremes Nuisance dusts/odors Baropressure extremes Fatigue Lifted weights Noise Vibration (Raynaud's syndrome) Mutagens Asphyxiants Allergens Pathogens Radiation (See Radiation) Cryogens Carcinogens Teratogens Toxins Irritants
Human Factors (See Ergonomics)	Operator error Inadvertent operation Failure to operate Operation early/late Operation out of sequence Right operation/wrong control

Category of hazard origination	Effect which may lead to loss
	Operated too long Operate too briefly
Ergonomic (See Human Factors)	Fatigue Inaccessibility Nonexistent/inadequate “kill” switches Glare Inadequate control/readout differentiation Inappropriate control/readout labeling Faulty work station design Inadequate/improper illumination
Unannunciated Utility Outages	Electricity Steam Heating/cooling Ventilation Air conditioning Compressed air/gas Lubrication drains/slumps Fuel Exhaust
Mission Phasing	Transport Delivery Installation Calibration Checkout Shake down Activation Standard start Emergency start Normal operation Load change Coupling/uncoupling Stressed operation Standard shutdown Shutdown emergency Diagnosis/troubleshooting Maintenance

E.2 Checklist for hazard sources

Following is a checklist from [83] of some general categories of hazard origination or source. Note that some of the factors are similar to those in Table 25, but are organized differently.

1. Acceleration
2. Contamination
3. Corrosion
4. Chemical dissociation
5. Electrical
   a. Shock
   b. Thermal (corresponds to electrical - overheating in Table 25)
   c. Inadvertent activation
   d. Power source failure (corresponds to electrical - power outage in Table 25)
   e. Electromagnetic radiation
6. Explosion
7. Fire

8. Heat and temperature
    a. High temperature
    b. Low temperature
    c. Temperature variations
9. Leakage
10. Moisture
    a. High humidity
    b. Low humidity
11. Oxidation
12. Pressure
    a. High
    b. Low
    c. Rapid change
13. Radiation
    a. Thermal
    b. Electromagnetic
    c. Ionizing
    d. Ultraviolet
14. Chemical replacement
15. Shock (mechanical)
16. Stress concentrations
17. Stress reveals
18. Structural damage or failure
19. Toxicity
20. Vibration and noise
21. Weather and environment

Following is a checklist of some categories of energy sources of hazards, assimilated from a variety of sources such as [83]:

1. Fuels
2. Propellants
3. Initiators
4. Explosive charges
5. Charged electrical capacitors
6. Storage batteries
7. Static electrical charges
8. Pressure containers
9. Spring-loaded devices
10. Suspension systems
11. Gas generators
12. Electrical generators
13. Radio frequency sources
14. Radioactive energy sources
15. Failing objects

16. Catapulted objects
17. Heating devices
18. Pumps, blowers, fans
19. Rotating machinery
20. Actuating devices
21. Nuclear

E.3 Checklist of hazard sources in Semiconductor Manufacturing

Table E.2 is a set of examples from the semiconductor manufacturing industry [84], organized by categories of sources of hazards and the corresponding potential loss or effect leading to potential loss.

Table 26: Checklist of hazard sources in semiconductor manufacturing equipment

Categories of hazard sources	Potential loss or effect which may lead to loss
Chemical Energy Chemical disassociation or replacement of fuels, oxidizers, explosives, organic materials or compounds	Fire Explosion Non-explosive exothermic reaction Material degradation Toxic gas production Corrosion fraction production
Contamination Producing or introducing contaminants to surfaces, orifices, filters, etc.	Clogging or blocking components Deterioration of fluids Degradation of performance sensors or operating components
Electrical Energy System or component potential energy release or failure. Includes shock, thermal, and static.	Electrocution/involuntary personnel reaction Personnel burns Ignition of combustibles Equipment burnout Inadvertent activation of equipment Release of holding devices Interruption of communications (facility interface) Electrical short circuiting
Human Hazards Human hazards including perception (inadequate control/display identification), dexterity (inaccessible control location), life support, and error probability (inadequate data for decision making). Conditions due to position (hazardous location/height), equipment (inadequate visual/audible warnings or heavy lifting), or other elements that could cause injury to personnel.	Personnel injury due to: Skin abrasion, cuts, bruises, burns, falls etc. Muscle/bone damage Sensory degradation or loss Death Equipment damage by improper operation/handling may also occur
Kinetic/Mechanical Energy (Acceleration) System/component linear or rotary motion. Change in velocity, impact energy of vehicles, components or fluids.	Impact Disintegration of rotating components Displacement of parts or piping Seating or unseating valves or electrical contact Detonation of shock sensitive explosives Disruption of metering equipment Friction between moving surfaces
Material Deformation Degradation of material due to an external catalyst (i.e., corrosion, aging, embrittlement, fatigue, etc.).	Change in physical or chemical properties; corrosion, aging, embrittlement, oxidation, etc. Structural failure De-lamination of layered material Electrical insulation breakdown
Natural Environment Conditions including lighting, wind, flood, temperature extremes, pressure, gravity, humidity, etc.	Structural damage from wind Equipment damage Personnel injury
Pressure System/component (e.g., fluid systems, air systems) potential energy including high, low, or changing pressure.	Blast/fragmentation from container over-pressure rupture Line/hose whipping Container implosion

Categories of hazard sources	Potential loss or effect which may lead to loss
	System leaks Aero-embolism, bends, choking, or shock Uncontrolled pressure changes in air/fluid systems
Radiation Conditions including electromagnetic, ionizing, thermal, or ultraviolet radiation (including lasers/and optical fibers).	Uncontrolled initiation of safety control systems & interlocks Electronic equipment interference Human tissue damage Charring of organic material Decomposition of chlorinated hydrocarbons into toxic gases Fuel ignition
Thermal High, low, or changing temperature	Ignition of combustibles Initiation of other reactions Expansion/contraction of solids or fluids Liquid compound stratification
Toxicants Inhalation or ingestion of substances by personnel	Respiratory system damage Blood system damage Body organ damage Skin irritation or damage Nervous system effects
Vibration/Sound System/component produced energy	Material failure Pressure/shock wave effects Loosing of parts Chattering of valves or contacts Verbal communications interference Degradation or failure of displays

E.4 Hazard sources in physical environment of an NPP DI&C safety system

Disruption in or emissions from the environment or physical conditions in the environment may degrade a safety function of the analyzed DI&C system in an NPP; e.g.:

1. Water in unwanted space
2. Transfer of unwanted energy in various forms; for example:
   - 2.1. Fire
   - 2.2. Lightening
   - 2.3. Heat
   - 2.4. Light
   - 2.5. Sound
   - 2.6. Vibration
   - 2.7. Radiation
   - 2.8. Shock
   - 2.9. Seismic event or effect
   - 2.10. Tsunami
   - 2.11. Flooding
   - 2.12. Electrostatic discharge
   - 2.13. Electromagnetic interference, causing spurious signal or signal change.
   - 2.14. Electromagnetic radiation, e.g.:
     - 2.14.1. Pulse
     - 2.14.2. Sunspot; solar flare
3. Interruption of services (primary; secondary; other forms of back-up) ; for example:
   - 3.1. Electric power supply.
4. Disturbance in services, propagating to a disturbance in a main signal; for example:

4.1. Electric power supply.
4.2. Service water [24]
4.3. Service air

5. Intrusions through breaches of isolation barriers; for example
   5.1. Cable penetration
   5.2. Other duct penetration
6. Adverse conditions in temperature, pressure, or humidity/moisture; for example
   6.1. Too high
   6.2. Too low
   6.3. Rapid changes
7. Disturbance in incoming signals
8. Misbehaving signals (data; commands) ; for example:
   8.1. Byzantine behavior.
   8.2. Behaving like a "babbling idiot" in a connected network.
9. Deprivation of resources; for example:
   9.1. Overloaded communication bus
   9.2. Resource locked up by other "users" of those resources.

Note: Items 8-9 are contributed through "logical" rather than physical sources in the environment.

E.5 Digital safety system contribution to hazards affecting its environment

Emissions or outputs from or behavior of the DI&C system having an effect on its environment may affect safety adversely; for example:

1. Emission of energy in various forms; for example:
   1.1. Heat
   1.2. Light
   1.3. Sound
   1.4. Vibration
   1.5. Electromagnetic radiation
   1.6. Electrostatic discharge.
2. Other unwanted, unplanned effluents, ; for example, those leading to
   2.1. Toxicity
   2.2. Inflammability
3. Output of signals (data; commands) ; for example:
   3.1. Byzantine behavior.
   3.2. Behaving like a "babbling idiot" in a connected network.
4. Excessive[132], load or demand on resources; for example:
   4.1. Electric power overload, due to a short circuit
   4.2. Communication bus overload
   4.3. Locking up resources, to the exclusion of other "users" of those resources.

[132] Excessive: Disruptive by exceeding limit declared or established in design.

Note: Items 3 and 4.2-4.3 are “logical” rather than physical contributory causes.

E.6 References for Appendix E

[82] NASA Reference Publication 1358, “System Engineering “Toolbox” for Design Oriented Engineers,” 1994.

[83] Ericson II., C.A., “Hazard Analysis Techniques For System Safety,” John Wiley and Sons, August 24, 2005.

[84] International SEMATECH, “Hazard Analysis Guide: A Reference Manual for Analyzing Safety Hazards on Semiconductor Manufacturing Equipment,” Technology Transfer # 99113846A-ENG, *November 30, 1999.*

Appendix F: Organizational qualities to support safety

Author: Dr. Dorothy Andreas, Pepperdine University
http://seaver.pepperdine.edu/academics/faculty/default.htm?faculty=dorothy_andreas

Integrative editing by Sushil Birla, Senior Technical Advisor for I&C, U.S. Nuclea Regulatory Commission

This appendix draws upon knowledge from the social sciences for the purpose of informing the evaluation of hazard analysis of a digital safety system for a nuclear power plant (NPP). Literature search in the social sciences did not yield any results specific to the context of engineering critical systems such as a digital safety system for a nuclear power plant (NPP). In the absence of context-specific research, this appendix assimilates[133] information from broader fields of applicable research, supporting the premise that collective mindfulness (Section F.5) within the organization is an essential factor for reducing the hazard space in engineering a digital safety system for an NPP, and for conducting the associated hazard analysis. Most of the scholarship is concerned with operations of technologically and organizationally intricate systems (nuclear power plants, aircraft carriers, aviation, petroleum industry, occupational safety, and healthcare) rather than the engineering of a digital instrumentation and control system [5], [19], [39]-[41], [44], [57]. Swanson, et al [43], theorizing about the application of HRO principles (Section F.1) to design of IT systems, is the only research that comes close to the context of RIL-1101 or the engineering of digital safety systems. Similarly, in this appendix we map the knowledge from the social sciences to the RIL-1101 context as follows:

> In the engineering environment, a high quality organization (HQEO) develops and maintains technological systems without entailing associated hazards, just as, in the operational environment, a "high reliability[134] organization" (HRO) operates hazardous technologies without leading to catastrophe" [46]-[47].

The subsequent sections describe specific behaviors and processes to develop collective mindfulness and discuss these in the context of accountability and standardization. Organizations can measure all of the factors described in the subsequent sections and use this information as one piece of evidence that a hazard analysis was performed, utilizing best communication practices and sound principles from the social sciences.

HQEOs, just as HROs, work hard to address intricacies within technical systems using processes that cultivate "collective mindfulness." Collective mindfulness is a set of stable cognitive processes that allow a group to develop sophisticated mental models that help to "improve hazard identification and evaluation" [46]-[47][135]. These organizations resist patterns of habitual[136] thinking and communicating that may lead them to miss safety-related information (e.g., contributors to hazards). They intentionally strengthen their collective ability to pay attention to new information to determine how the information provides insight into the intricacies of the system and help the organization avoid a hazardous condition[137] and prevent the consequential loss (e.g., degradation of a safety function). Organizational culture is a

[133] Assimilation includes mapping certain terms to the context of RIL-1101.

[134] Reliability in this context does not have the same definition as used in fault-tolerant engineered systems.

[135] It is implicit in the expression, "risk detection, assessment and management" in the cited references.

[136] Interpreting new information through an old reference frame - the traditional belief system.

[137] The reference uses the terms, "error, failure"

contributing factor to individuals' abilities to develop collective mindfulness. There are also specific communication behaviors that enable organizations to develop collective mindfulness.

F.1 Five Principles

Following are the five principles to organize for high quality[138] processes [44], [46]-[47][72]:

1. Preoccupation with hazard identification[139] — treat every piece of information as a potential symptom that something could be wrong with the system.
2. Reluctance to accept simplistic[140] explanations and models — always hold current mental models in question with a persistent goal to create more complete and nuanced[141] explanations and models of the system.
3. Sensitivity to operations — situational awareness of the (current state of the) system — be able to notice anomalies[142], track them, and resolve them.
4. Commitment to resilience — learning from mistakes, correcting[143] their perceptions to represent reality well enough to identify (contributory) hazards their perceptions — detect, contain, and recover from mistakes[144].
   4.1. Ability to respond to unanticipated conditions (outside the boundary of the organization's deterministic processes) without compromising its end goal[145].
   4.2. Ability to learn and grow from previous episodes of resilient action.
5. Deference to expertise — cultivate diversity and delegate (empower) people, who are closer to the situation and can recognize more subtle contributors to a hazard in intricate environments and assimilate information from their diverse perspectives.

HROs generally practice these principles in their everyday activities. However, there are ways to measure an organization's ability to follow their principles with surveys. A survey measure of the five principles is in [46]-[47].

F.2 Accountability, Standardization, and Adaptation

Many assumptions associated with safety management are based in traditional "scientific" management, inheriting the following characteristics:

---

[138] The references use the term, "reliability"

[139] The references use the term "failure."

[140] In the RIL-1101 context, it means "not adequately representative of reality, missing (contributory) hazards"

[141] In the RIL-1101 context, it means "reflecting subtle details that enable (contributory) hazard identification."

[142] In the RIL-1101 context, it maps into "(contributory) hazards."

[143] The references use the term "complicating."

[144] The cited reference uses the terms, "failure" and "error" for which "unrecognized hazard" is the corresponding concept in RIL-1101. Its effect may be an "unwanted loss" for which, in the context of organizational processes, the cause is traced to some mistake by some human.

[145] The cited reference uses the expression, "absorb strain and preserve functioning despite presence of adversity"

1. A standardization[146] of work process, output, skills, and organization norms (e.g. safety culture).
2. A strict separation of planning[147] and operations processes.
3. The use of "scientific" measurement to develop the standardization and to detect flaw in the system [3].

The basic assumption of "scientific" management is that standardized processes in normal operations control output and prevent mistakes [3], [50]. This fundamental assumption is related to master premises that efficiency and predictability are desirable performance characteristics[148] of organizational processes [37], [50]. In the context of safety management, it assumes that managers can control employee behavior and that mishaps result from performance shortfalls, which are the product of failing to control employee behavior (e.g. "mistakes") [3], [37]. The core assumptions of "scientific" management do not make adequate provision[149] for unanticipated conditions and limit their ability to recognize (contributory) hazards [3], [31], [37], [46]-[47].

Likewise, the desire to establish a clear hierarchical "command and control" tree derives from these assumptions [50]-[51]. Decades of research about organizations, including the nuclear industry, clearly document that the very nature of bureaucracy in organizations diffuses[150] accountability [50]. In some ways this is a strong point of bureaucracy because their prescriptive "deterministic" processes enable accomplishment of organizational tasks and goals without full dependence on an individual thinking for adjusting to situation-specific unanticipated conditions [50]—thus, individuals often base decisions on the assumptions, underlying the "deterministic" processes, but not always made explicit [50]. However, as noted in Table 2, H-culture-9, an overly rigid "command and control" organization structure can increase the hazard space because the implicit assumptions and premises may not hold.

Organizational research asserts that the nature of bureaucracy creates a powerful force to diffuse accountability throughout the organization [50]- [51]. In terms of ethics, some researchers lament this organizational force and call organizations, in general, to become mindful of this tendency and counteract it whenever possible [50]. But rather than tracing all decisions through individual accountability, they suggest that organizational members question assumptions and premises that pervade the organizational culture [50]. The Toulmin model introduced in Appendix C.3.3 is one technique by which organizational members can question premises and assumptions as they relate to evidence and claims about hazards or hazard control. Conversations that seek to make these elements of arguments transparent can help counteract the diffusion of accountability in bureaucracy. Of course, the intricacy of these conversations and the amount of information that must be considered in hazard analysis can make it difficult to keep a record of deliberations, decisions, and rationale. Knowledge management tools such as dialogue mapping can help organizations keep track of deliberations, decisions, and rationale and hyperlink the rationale with supporting information and documents [54]-[56].

---

[146] It includes top-down decomposition and allocation of responsibilities along the organization (command and control) structure, down to the individual.

[147] Rigid hierarchical (top-down) plans limit local autonomy during execution or operation.

[148] The references use the term, "outcomes."

[149] Example: Organizational architecture for collective mindfulness.

[150] The top-down allocation of roles, responsibilities and performance metrics does not make adequate provision for bottom-up observation and feedback of real conditions and adaptation to cope with associated contributory hazards..

Even though the use of Toulmin's argumentation model can help counteract diffusion of authority in organizations, a caveat is in order in the context of complex, high-risk technology. One of the main goals of Perrow's Theory of Normal Accidents [68] is to raise awareness of the faulty assumption that accidents results from a lapse of "scientific" management to control employees — often referred to as "human error" [68], [70]. In the context of complex, high-risk technologies, it is worth considering his argument that the nature of complex technical systems makes it extraordinarily complex for standardization of organizational procedures to anticipate all possible combinations of mistakes. HROs take this issue seriously by developing collective mindfulness in order to create requisite diversity and independence in the organizational system to recognize the complexity of the technical system [39], [46]-[47]. Requisite variety is the variation in frames of reference and knowledge that makes the organization capable of recognizing and addressing hazards [44]. In the case of many organizational mishaps, the paradox is that the standardization of process that was designed to control mistakes, in fact, minimized the organizations' ability to develop collective mindfulness that would prevent the mishap [31]-[32], [69]-[70]. Alternately, HRO-relevant research in nuclear power plants, aircraft carriers, aviation, and the petroleum industry consistently demonstrates that these organizations centralize and standardize procedures while also building collective mindfulness about when to decentralize[151] and adapt the procedures [46]-[47], [50]. It is also important to note that too much emphasis on the separation of planning and execution can lower the organization's collective mindfulness because it lowers sensitivity to the context and the system [2]-[3], [50], [58].

Thus, the desire to develop accountability and standardization within organizations must be accomplished without minimizing the organizations' ability to develop collective mindfulness that allows them to recognize and prevent (contribution to) hazards. The subsequent sections discuss the relationship between organization culture and decisional premises (Section F.3), the role of communication for developing collective mindfulness and following Toulmin's model of argumentation (Section F.4), and the relationship between professional identification and collective mindfulness and competence (Section F.5). Additionally, each section references tools and techniques of measuring the organizational and communication factors.

F.3 Organizational culture and decisional premises

The organization's culture can create values and decision premises that guide individual members' cognition, communication, and processes in a manner that increases safety [8], [58]-[59], [71]. Organizational culture is a complex concept, and due to its complexity, it is difficult to define conceptually and difficult to measure [8], [17], [33], [37]. Following is the most commonly cited definition of organizational culture: "Organizational culture is understood to be deeply rooted assumptions about human nature, human activities, and social relationship shared by members of an organization and their expression in values, behavioral patterns, and artifacts found within the organization" [71].

In the nuclear industry (and others), this concept is often called safety culture defined by IAEA as "that assembly of characteristics and attitudes in organizations and individuals, which establishes that, as an overriding priority, nuclear plant safety issues receive the attention warranted by their significance" [59]. Thus, one important way to think about the role of organizational culture in the process of hazard analysis is members of the organization would be motivated by their value of safety to pay close attention to hazard-related information.

[151] Delegate and distribute control; provide the autonomy (empower) to adapt, learn, and feedback.

In addition to establishing core values of an organization, the culture carries premises and assumptions that often become the basis for decisions and evaluation of information in the organization. It is the HRO's established premises that allow it to have centralized and standardized processes while at the same time allowing members interpretive flexibility to recognize new information and adapt work processes accordingly [5], [58].

The discipline of organizational culture derives from an anthropological tradition of studying culture and organizations. It examines patterns of meaning, values, and frames of references that are shared among members of a community. It considers culture to be a complex whole of knowledge, beliefs, ethics, and customs that is both created and lived within members of a community. These cultural frames of reference are the lenses through which community members interpret and evaluate information and behavior. Given the complexity and dynamic nature of organizational culture, it is a very complex phenomenon to measure. It is best evaluated with a combination of qualitative and quantitative measures. There are many 3-part frameworks to measure organizational culture. One framework suggests that it is a dynamic interrelationship between individual characteristics, behavior, and the environment [60]. A similar model suggests that individual behavior is influenced by the triad of organizational structure, organizational processes, and organizational culture [17]. Qualitative measures might include themes and patterns from a series of employee interviews, thematic analysis of focus groups, detailed observation of the work environment, and audits of organizational documents. Another approach uses rubrics to assess five levels of safety culture: (1) organization does not care about safety, (2) organization increases safety after an accident, (3) organization uses systems and procedures to prevent hazards, (4) organization tries to anticipate safety problems, and (5) normalization of safety values within the organization culture (akin to the principles of highly reliable organizations). Even though these measures of safety provide a sense of the values and interpretive frames within a community, it is important to recognize that any measure only captures a moment in time and does not tell the entire story.

There have been many efforts to develop quantitative measures of safety culture. These efforts are generally considered to be measuring safety climate. Safety climate is an aggregation of individual attitudes of safety. Thus, safety climate measured in surveys is a manifestation of some aspect of the organizational safety culture. Even given this qualification of a survey approach, many scholars question the validity of these surveys and suggest they are simply measuring employee satisfaction with the organization and their supervisors [17]. Thus, reports of survey-measures should be evaluated carefully.

One approach to measuring safety culture suggests that the organization should carefully consider what it really wants to measure [10]. One question interrogates the organizational culture as an attribute of the organization--as something the organization has. Measurement methods appropriate to this question include observation and audits. A second question interrogates how the organizational culture impacts individual attitudes about safety. Measurement methods for the second question include surveys and observation. A third question interrogates the organizational climate as seen through the eyes of employees, contractors, and external audiences. Measurement methods for the third question include interviews and surveys. This approach suggests that technique of measuring organizational safety culture should be based on reason (purpose) for measuring it.

See [2] for opinions about: incident reporting, manager, prioritization of worker safety, work procedures, work situation and stress, competence and training, communication and cooperation, upper management, lines of responsibility, and perceptions of vocation (in this case seamanship).

See [10] for attitudes toward management commitment to safety, priority of safety, communication, safety rules, supportive environment, involvement, personal priorities and need for safety, personal appreciation of risk, and nature of work.

F.4 Communication for collective mindfulness

Quality of hazard analysis is affected by the quality of interaction among the involved people. Good interaction quality depends upon individual communication competence (Section F.4.1), participatory communication climate (Section F.4.2), cross-disciplinary or interdisciplinary competence (Section F.4.3), and prevention of GroupThink (Section F.4.4).

F.4.1 About Becoming a Competent Communicator

In general, the field of Communication Studies has given considerable thought to the qualities of a competent communicator. Even though there are many lively debates about this topic, most scholars accept the fundamental assumption that competent communicators effectively manage three goals: (1) to present a competent and credible image of self, (2) to escalate, maintain, or terminate relationships, and (3) to accomplish instrumental tasks [53]. The research about group communication and inter-disciplinary communication indicates that sole focus on the third goal, ignoring the other two goals, increases the hazard space and prevents organizations from developing collective mindfulness. Thus, the assumption about competence communicators managing these three goals pervades the subsequent discussions.

One commonly referenced model of communication competence identifies six factors of communication competence, measurable with a survey [53]:

1. Ability to adapt communication to the context.
2. Ability to stay cognitively involved in the conversation and to demonstrate involvement with appropriate verbal and non-verbal cues.
3. Ability to manage a conversation effectively through turn-taking, questioning, intonation, topic shifts, extensions etc.
4. Ability to understand a person's perspective and emotions.
5. Ability to achieve the goal of the conversation.
6. Ability to uphold social norms and expectations for what counts as appropriate for a given situation.

F.4.2 Participatory Communication Climate

A participatory communication climate at an organization contributes to the organization's ability to follow the five HRO principles and develop collective mindfulness. There are four characteristics of participatory communication climate that contribute to collective mindfulness: (1) that individuals have voice to express ideas and concerns, (2) that the organization has an open communication climate, (3) that individuals have easy access to relevant information, and (4) individuals engage in continuous and ongoing learning. These characteristics of participatory communication climate are measured with a survey published in [30].

F.4.3 Collective Communication Competence and Diversity

Communication among individuals from various professional and disciplinary backgrounds has the potential to increase intellectual diversity and this is a factor that contributes to collective mindfulness [30], [46]-[47]. Unfortunately, interdisciplinary communication is also challenging.

In particular, the following communication activities are a contributory hazard because they limit organizations' ability to develop inter-disciplinary competence:

1. Expressions of negative humor and sarcasm,
2. Debating with team members about whose expertise is more important and jockeying for control and power,
3. Expressing boredom through verbal and nonverbal messages [49].

Some of the items may seem as though they are addressing organizational minutiae; however, it is worth examining these behaviors, because, if it is too frequent, these behaviors limit an organization's ability to seek and use intellectual diversity for recognizing hazards.

Teams can increase intellectual diversity by developing collective communication competence of interdisciplinary group communication. The following behaviors increase collective communication competence [49]:

1. Building trusting relationships.
2. Reflectively talking about the task when members spend time coordinating their understanding of what to do (this is related to Steps #1 and #2 of group conversational quality in Section F.4.4).
3. Negotiating meaning by discussing different uses of language that arise from disciplinary and professional differences (this would be especially important as nuclear engineers collaborate with software engineers).
4. Demonstrating presence through active listening behaviors.
5. Informal communication, such as shared humor, that builds positive relationships and sense of shared meaning.

Through these behaviors, individuals can meet all three goals that are related to communication competence (see Section F.4.1).

F.4.4 Conversation Quality and Deference to Expertise

Groupthink is an organizational phenomenon that leads to poor quality decisions and increases the hazard space [61]. Groupthink occurs when group members feel a strong sense of cohesiveness.

F.4.4.1 Characteristics of GroupThink

Six characteristics of groupthink have been identified as follows [48], [61]:

1. Critical thinking is not encouraged or rewarded.
2. Members of the group are so cohesive that they believe they can do no wrong.
3. Members are too focused on justifying their own actions.
4. Members often believe that they have reached a true consensus.
5. Members are too concerned with reinforcing the leader's beliefs and attitudes.

Groupthink is a contributory hazard because it limits the organization's ability to develop collective mindfulness. In the context of hazard analysis of digital safety systems, it can diminish the organization's ability to be deferent to expertise across the many relevant contexts.

F.4.4.2 Countermeasures to prevent GroupThink

In order to counter the possibility of groupthink, groups can develop quality conversations that lead to high quality decisions (or in the context of RIL-1101, high quality hazard analysis of digital instrumentation and controls).

Five conversational acts[152] that can improve conversational quality for hazard analysis have been identified [62]-[64]:

1. Carefully gather information to identify a hazard; analyze the information in a way that results in a clearly defined hazard.
2. Set criteria for the quality of the decision about this hazard; examples:
   2.1. Explicit articulation of premises and assumptions [67]
   2.2. Preventing diffusion of accountability in the organization (see Section F.2)
   2.3. Group conversational quality can be measured using the Competent Group Communicator Scale [48].
3. Identify factors to reduce the hazard space; seek a range of constraint alternatives.
4. Critically evaluate the identified hazard (act 1), and the alternatives to reduce the hazard space (act 3).
5. Select the best course of action to avoid, eliminate or otherwise control the hazard; remain open to new information; be willing to revise as needed.

F.5 Collective mindfulness and competence

A survey measure of collective mindfulness is in [4], [30].

F.6 Glossary for Appendix F

Accountability

The quality or state of being accountable (responsible).

Cultivate

Develop (improve) a pattern of behavior.

Cognitive process

The performance of some composite cognitive activity; an operation that affects mental contents.

Collective mindfulness

A characteristic of an organization of having cognition, the collective mindset, necessary to detect and understand unanticipated conditions[153] and for recovery before they lead to harm.

---

[152] These five conversational acts are modified to adjust to the context of hazard analysis. In the research, the five acts contribute to a high quality decision: (1) define the problem, (2) set criteria for a solution, (3) propose possible solutions, (4) critically evaluate proposals, (5) select the best proposal.

[153] In the context of RIL-1101, these are mapped into "(contributory) hazards."

Note: Awareness is more than simply an issue of "the way in which scarce attention is allocated." Mindfulness is as much about the quality of attention as it is about the conservation of attention. It is as much about what people do with what they notice as it is about the activity of noticing itself. Mindfulness involves interpretive work directed at weak signals, differentiation of received wisdom, and reframing, all of which can enlarge what is known about what was noticed. It is the enlarged set of possibilities that suggests unexpected deviation[154] that needs to be corrected and new sources of ignorance that become new imperatives for noticing.

Complex; complexity

Note: See in Appendix A

High Reliability Organization (HRO)

Organization that operates (works with) hazardous (hazard-contributing) technologies without leading to catastrophe (loss of safety).

Organizational culture

Deeply rooted assumptions about human nature, human activities, and social relationship shared by members of an organization and their expression in values, behavioral patterns, and artifacts found within the organization.

F.7 References for Appendix F

[1] Antonsen, Stian, Safety Culture and the Issue of Power, Safety Science, Vol. 47, No. 2, 2009a.
[2] The Relationship Between Culture and Safety on Offshore Supply Vessels, Safety Science, Vol. 47, No.8, 2009b.
[3] Antonsen, Stian, Skarholt, Kari, Ringstad, Arne Jarl, The Role of Standardization in Safety Management – A Case Study of a Major Oil & Gas Company, Safety Science, Vol. 50, No. 10, 2012.
[4] Barrett, M. Scott, Novak, Julie M., Venette, Steven J., Shumate, Michelle, Validating the High Reliability Organization Perception Scale, Communication Research Reports, Vol. 23, No. 2, 2006.
[5] Bierly III, Paul E., J.-C., Spender, Culture and High Reliability Organizations: The Case of the Nuclear Submarine, Journal of Management, Vol. 21, No. 4, 1995.
[6] Carvalho, Paulo V. R., dos Santos, Isaac L., Gomes, José Orlando, Borges, Marcos R. S., Micro Incident Analysis Framework to Assess Safety and Resilience in the Operation of Safe Critical Systems: A Case Study in a Nuclear Power Plant, Journal of Loss Prevention in the Process Industries, Vol. 21, No. 3, 2008.
[7] Carvalho, Paulo V. R., dos Santos, Isaac L., Vidal, Mario C. R., Nuclear Power Plant Shift Supervisor's Decision Making During Microincidents, International Journal of Industrial Ergonomics, Vol. 35, No. 7, 2005.
[8] Choudhry, Rafiq M., Fang, Dongping, Mohamed, Sherif, The Nature of Safety Culture: A Survey of the State-of-the-Art, Safety Science, Vol. 45, No. 10, 2007.
[9] Cooper Ph.D., M. D., Towards a Model of Safety Culture, Safety Science, Vol. 36, No. 2, 2000.
[10] Cox, S. J., Cheyne, A. J. T., Assessing Safety Culture in Offshore Environments, Safety Science, Vol. 34, pp. 111–29, 2000.
[11] Le Coze, Jean Christophe, Towards a Constructivist Program in Safety, Safety Science, Vol. 50, No. 9, 2012.

[154] In the context of RIL-1101, deviation is mapped into "(contributory) hazard."

[12] What Have We Learned About Learning from Accidents? Post-Disasters Reflections, Safety Science, Vol. 51, No. 1, 2013.
[13] Denyer, David, Kutsch, Elmar, Lee-Kelley, Elizabeth (Liz), Hall, Mark, Exploring Reliability in Information Systems Programmes, International Journal of Project Management, Vol. 29, No. 4, 2011.
[14] Díaz-Cabrera, D., Hernández-Fernaud, E., Isla-Díaz, R., An Evaluation of a New Instrument to Measure Organisational Safety Culture Values and Practices, Accident Analysis and Prevention, Vol. 39, No. 6, 2007.
[15] Fernández-Muñiz, Beatriz, Montes-Peón, José Manuel, Vázquez-Ordás, Camilo José, Safety Culture: Analysis of the Causal Relationships Between Its Key Dimensions, Journal of Safety Research, Vol. 38, No. 6, 2007
[16] Flin, R., Mearns, K., Connor, P. O., Bryden, R., Measuring Safety Climate: Identifying the Common Features, Safety Science, Vol. 34, pp. 177–92, 2000.
[17] Guldenmund, Frank W., The Use of Questionnaires in Safety Culture Research – An Evaluation, Safety Science, Vol. 45, No. 6, 2007.
[18] Hale, A.R., Guldenmund, F.W., van Loenhout, P.L.C.H., Oh, J.I.H.., Evaluating Safety Management and Culture Interventions to Improve Safety: Effective Intervention Strategies, Safety Science, Vol. 48, No. 8, 2010.
[19] Hofmann, David A., Jacobs, Rick, Landy, Frank, High Reliability Process Industries: Individual, Micro, and Macro Organizational Influences on Safety Performance, Journal of Safety Research, Vol. 26, No. 3, 1995.
[20] Hopkins, Andrew, Studying Organisational Cultures and Their Effects on Safety, Safety Science, Vol. 44, No. 10, 2006.
[21] Lee, T., Harrison, K., Assessing Safety Culture in Nuclear Power Stations, Safety Science, Vol. 34, No. 1-3, 2000.
[22] Mariscal, M. A., García Herrero, S., Toca Otero, A., Assessing Safety Culture in the Spanish Nuclear Industry through the Use of Working Groups, Safety Science, Vol. 50, No. 5, 2012.
[23] Martínez-Córcoles, Mario, Gracia, Francisco J., Tomás, Inés, Peiró, José M., Schöbel, Markus, Empowering Team Leadership and Safety Performance in Nuclear Power Plants: A Multilevel Approach, Safety Science, Vol. 51, No. 1, 2013.
[24] Martínez-Córcoles, Mario, Gracia, Francisco, Tomás, Inés, Peiró, José M., Leadership and Employees' Perceived Safety Behaviours in a Nuclear Power Plant: A Structural Equation Model, Safety Science, Vol. 49, No. 8–9, 2011.
[25] McFadden, Kathleen L., Henagan, Stephanie C., Gowen III, Charles R., The Patient Safety Chain: Transformational Leadership's Effect on Patient Safety Culture, Initiatives, and Outcomes, Journal of Operations Management, Vol. 27, No. 5, 2009.
[26] Mitropoulos, Panagiotis "Takis," Cupido, Gerardo, The Role of Production and Teamwork Practices in Construction Safety: A Cognitive Model and an Empirical Case Study, Journal of Safety Research, Vol. 40, No. 4, 2009.
[27] Nævestad, Tor-Olav, Evaluating a Safety Culture Campaign: Some Lessons from a Norwegian Case, Safety Science, Vol. 48, No. 5, 2010.
[28] Navarro, M., Latorre, Felisa, Gracia Lerín, Francisco J., Tomás, Inés, Peiró Silla, José María, Validation of the Group Nuclear Safety Climate Questionnaire, Journal of Safety Research, Vol. 46, No. 0, 2013.
[29] Neal, A., Gri, M. A., Hart, P. M., The Impact of Organizational Climate on Safety Climate and Individual Behavior, Vol. 34, pp. 99–109, 2000.
[30] Novak, Julie M., Sellnow, Timothy L., Reducing Organizational Risk through Participatory Communication, Journal of Applied Communication Research, Vol. 37, No. 4, 2009.

[31] Pidgeon, N., O'Leary, M., Man-Made Disasters: Why Technology and Organizations (Sometimes) Fail, Safety Science, Vol. 34, No. 1-3, 2000.
[32] Pidgeon, Nick, The Limits to Safety ? Culture, Politics, Learning and Man-Made Disasters, Journal of Contingencies and Crisis Management, Vol. 5, No. 1, 1997.
[33] Safety Culture: Key Theoretical Issues, Work & Stress, Vol. 12, No. 3, 1998.
[34] Rao, Suman, Safety Culture and Accident Analysis—A Socio-Management Approach Based on Organizational Safety Social Capital, Journal of Hazardous Materials, Vol. 142, No. 3, 2007.
[35] Reiman, T., Oedewald, P., Measuring Maintenance Culture and Maintenance Core Task with CULTURE-Questionnaire—A Case Study in the Power Industry, Safety Science, Vol. 42, No. 9, 2004.
[36] Reiman, Teemu, Oedewald, Pia, Assessing the Maintenance Unit of a Nuclear Power Plant – Identifying the Cultural Conceptions Concerning the Maintenance Work and the Maintenance Organization, Safety Science, Vol. 44, No. 9, 2006.
[37] Assessment of Complex Sociotechnical Systems – Theoretical Issues Concerning the Use of Organizational Culture and Organizational Core Task Concepts, Safety Science, Vol. 45, No. 7, 2007.
[38] Reiman, Teemu, Oedewald, Pia, Rollenhagen, Carl, Characteristics of Organizational Culture at the Maintenance Units of Two Nordic Nuclear Power Plants, Reliability Engineering & System Safety, Vol. 89, No. 3, 2005.
[39] Rijpma, Jos A., Complexity, Tight-Coupling and Reliability: Connecting Normal Accidents Theory and High Reliability Theory, Journal of Contingencies and Crisis Management, Vol. 5, No.1, 1997.
[40] Roberts, Karlene H., Rousseau, Denise M., La Porte, Todd R., The Culture of High Reliability: Quantitative and Qualitative Assessment Aboard Nuclear-Powered Aircraft Carriers, The Journal of High Technology Management Research, Vol. 5, No. 1, 1994.
[41] Skjerve, A. B., The Use of Mindful Safety Practices at Norwegian Petroleum Installations, Safety Science, Vol. 46, No. 6, 2008.
[42] Skjerve, A.B., Kaarstad, M., Størseth, F., Wærø, I, Grøtan, T. O., Planning for Resilient Collaboration at a New Petroleum Installation: A Case Study of a Coaching Approach, Safety Science, Vol. 50, No. 10, 2012.
[43] Swanson, E. Burton, Neil, C., Innovating Mindfully with IT Technology, MIS Quarterly, Vol. 28, No. 4, 2004.
[44] Weick, Karl E., Sutcliffe, Kathleen M., Obstfeld, David, Organizing for High Reliability: Processes of Collective Mindfulness, In Crisis Management Volume III, ed. Boin, Arjen, Thousand Oaks, CA: Sage, 2008.
[45] Ziegler, Jennifer A., The Story Behind an Organizational List: A Genealogy of Wildland Firefighters' 10 Standard Fire Orders, Communication Monographs, Vol. 74, No. 4, 2007.
[46] Weick, K. E., Sutcliffe, K. M., Managing the Unexpected: Resilient Performance in an Age of Uncertainty, San Francisco, CA: John Wiley and Sons, 2007.
[47] Weick, K. E., Roberts, K. H., Collective Mind in Organizations: Heedful Irrelating on Flight Decks, Administrative Science Quarterly, Vol. 38, No. 3, 1993.
[48] Beebe, S. A., Masterson, J. T., Communicating in Small Groups: Principles and Practices (8th ed.), Boston, MA: Allyn & Bacon, 2004.
[49] Thompson, J. L., Building Collective Communication Competence in Interdisciplinary Research Teams, Journal of Applied Communication Research, Vol. 37, No. 3, 2009.
[50] Cheney, G., Christensen, L. T., Zorn, T., Ganesh, S., Organizational Communication in an Age of Globalization: Issues, Reflections, Practices (2nd ed.), Long Grove, IL: Waveland Press, n.d.
[51] Simon, Herbert, Administrative Behavior, 4th edition, New York: Free Press, 1997.

[52] Holford, W. D., Knowledge Construction and Risk Induction/Mitigation in Dialogical Workgroup Processes, Qualitative Research in Organizations and Management: An International Journal, Vol. 5, No. 2, 2010.
[53] Canary, D. J., Cody, M. J., Manusov, V., Interpersonal Communication: A Goals-Based Approach, Boston, MA: Bedford St. Martin's, 2003.
[54] Bracewell, R. H., Wallace, K. M., Moss, M., Knott, D., Capturing Design Rationale, Computer-Aided Design, Vol. 41, No. 3, 2009.
[55] Eng, N. L., Aurisicchio, M., Bracewell, R. H., Armstrong, G., More Space to Think: Eight Years of Visual Support for Rationale Capture, Creativity and Knowledge Management in Aerospace Engineering, In DETC/CIE, pp. 1–11, Washington, DC: ASME, 2011.
[56] Conklin, J., Dialogue Mapping: Building Shared Understanding of Wicked Problems, Chichester, UK: John Wiley & Sons, 2006.
[57] Schulman, P. R., The Negotiated Order of Organizational Reliability, Administration and Society, Vol. 25, No. 3, 1993.
[58] Weick, Karl E., Organizational Culture as a Source of High Reliability, California Management Review, Vol. 29, No. 2, 1987.
[59] International Nuclear Safety Advisory Group (INSAG), Basic Safety Principles for Nuclear Power Plants (Safety Series No 75-INSAG-3), International Atomic Energy Agency, Vienna, 1988.
[60] Geller, E. S., Ten Principles for Achieving a Total Safety Culture, Professional Safety, pp. 18-24, 1994.
[61] Janis, I. L., Crucial Decisions: Leadership in Policymaking and Crisis Management, New York: The Free Press, 1989.
[62] Orlitzky, M., Hirokawa, R. Y., The Err is Human, to Correct for it Divine: A Meta-Analysis of Research Testing the Functional Theory of Group Decision-Making Effectiveness, Small Group Research, Vol. 32, pp. 313-341, 2001.
[63] Gouran, D. S., Hirokawa, R. Y., Effective Decision Making and Problem Solving: A Functional Perspective, In Hirokawa, R. Y., Cathcart, R. S., Samovar, L. A., Henman, L. D., (Eds.), Small Group Communication Theory and Practice: An Anthology (8th ed), Los Angeles: Roxbury, 2003.
[64] Wittenbaum, G. M., Hollingshead, A. B., Paulus, P. B., Hirokawa, R. Y., Ancona, D. G., Peterson, R. S., Jehn, K. A., Yoon, K., The Functional Perspective as a Lens for Understanding Groups, Small Group Research, Vol. 35, pp. 17-43, 2004.
[65] Foss, S. K., Foss, K. A., Trapp, R., Contemporary Perspectives on Rhetoric, 3rd edition, Prospect Heights, IL: Waveland, 2002.
[66] Bier, V. M., On the State of the Art: Risk Communication to the Decision-Makers, Reliability Engineering and System Safety, Vol. 71, pp. 151–157, 2001.
[67] Toulmin, Stephen, The Uses of Argument, Cambridge: University Press, 1958.
[68] Perrow, C., Normal Accidents: Living with High-Risk Technologies, Princeton, NJ: Princeton University, 1999.
[69] Vaughn, D., The Challenger Launch Decision: Risky Technology, Culture, and Deviance and NASA, Chicago, IL: University of Chicago Press, 1996.
[70] Perin, Constance, Shouldering Risk: The Culture of Control in the Nuclear Power Industry, Princeton, NJ: Princeton University Press, 2006.
[71] Schein, E., Organizational Culture and Leadership, San Francisco, CA: Wiley and Sons, 2010.
[72] URL: http://witandwisdomofanengineer.blogspot.com/2011/05/failure-of-imagination.html

Appendix G: Example case studies

These cases studies illustrate how much can be learned from a single event to prevent or avoid a broader range of mishaps. When a specific mishap is examined for its causes (contributory hazards), pre-existing knowledge of cause-effect relationships can be used as the basis for generalizing from the specific contributory occurrences to more general contributory hazards.

The concept of generalization has been used in a systems engineering process, where a set of scenarios are used (in addition to general requirements) to imply and represent many similar situations, conditions, and cases; these scenarios drive the engineering of the system. The resulting system not only satisfies the requirements explicit in the scenarios, but also many other implied scenarios.

Experts [85] in such generalization have identified two types of reasoning processes, abduction and induction.

G.1 Ft Calhoun Event

Following is an excerpt from the “Ft Calhoun Oversight Increase Dec 13 announcement” [86] and the Fort Calhoun Station Inspection Report [87].

The plant was shut down on April 9 for a refueling outage. The outage was extended due to flooding along the Missouri River. Then an electrical fire on June 7 led to the declaration of an “Alert” and caused further restart complications.

The fire had resulted in the loss of spent fuel pool cooling capability for a brief time and caused significant unexpected system interactions.

The Alert caused by the (electrical circuit) breaker fire resulted from inadequate design or installation of electrical components. Deficiencies were noted with environmental qualification analyses for plant structures, systems and components. These analyses are relied on to demonstrate that key systems will be able to perform their safety functions under a variety of challenging accident conditions like earthquakes, loss of coolant accidents, high radiation fields, seismic events, etc.

Figure 13 illustrates the causality relationships extracted from the textual information above. It illustrates a generalization from the specific occurrence in Ft Calhoun. In this example, the deficiency in the component design was not caught in the V&V activities. However, if we survey known causes of “deficient designs”, the leading cause is “deficient requirements.” Experience in software-reliant systems for many application domains has consistently shown this to be the leading cause. In the context of RIL-1101, “deficient requirements” implies inadequate HA (e.g., inadequate understanding of contributory hazards; inadequate formulation of requirements to avoid or prevent such contributory hazards, and inadequate validation of the HA and the resulting requirements).

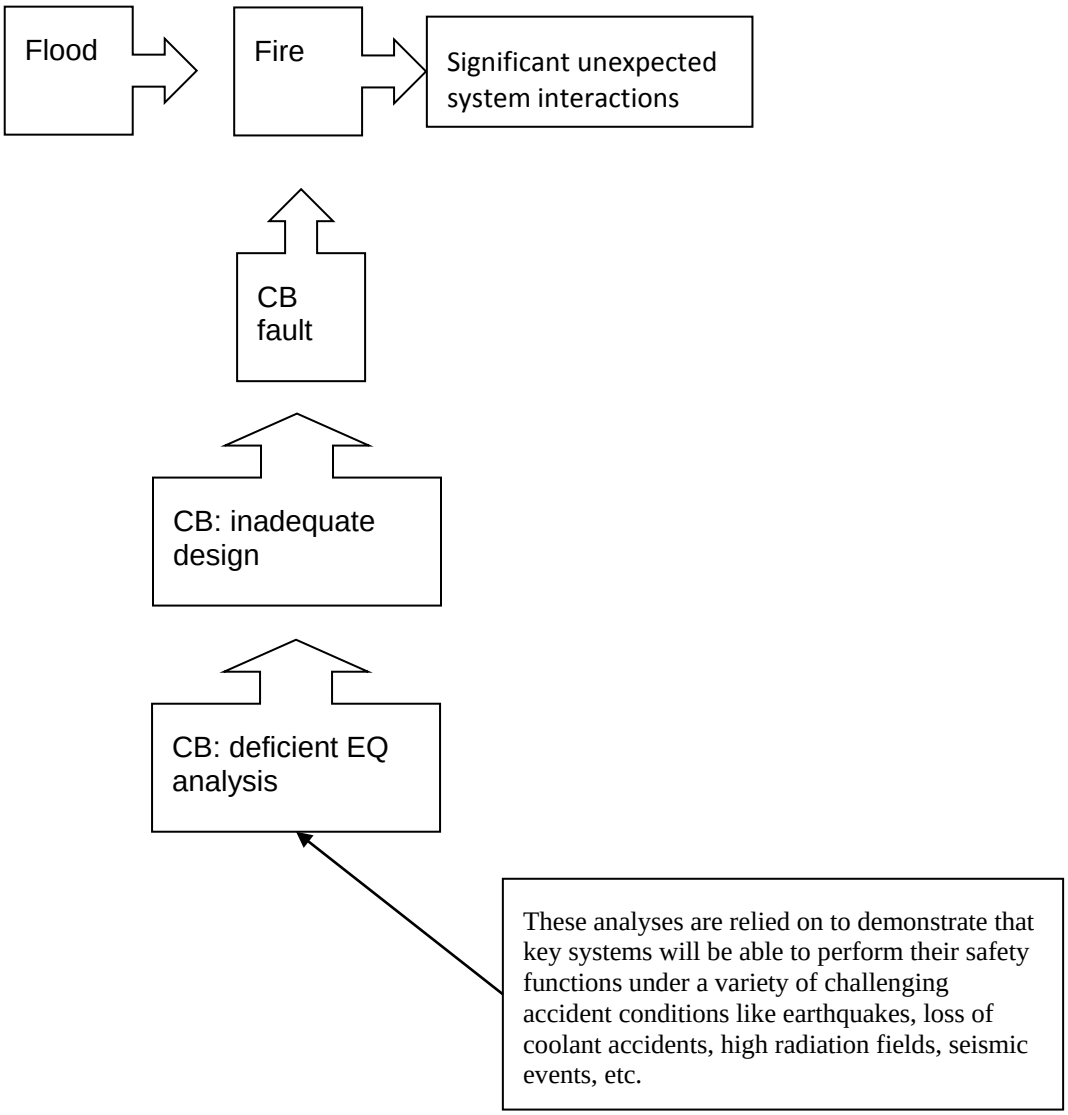


Figure 13: Example from event on June 7, 2011 at Ft Calhoun NPP

> "The power of generalizing ideas, of drawing comprehensive conclusions from individual observations, is the only acquirement, for an immortal being, that really deserves the name of knowledge. "Mary Wollstonecraft (1759–1797), British feminist. A Vindication of the Rights of Woman, ch. 4 (1792)." [88]

References for Appendix G

[85] Abduction and Induction – Essays on their relation and integration, Kluwer Academic Publishers, ISBN 0-7923-6250-0, editors Peter A. Flach and Antonis C. Kakas, 2000

[86] U.S. Nuclear Regulatory Commission, "Inadequate Flooding Protection Due to Ineffective Oversight," Licensee Event Report 285-2011-003, May 1, 2011.

[87] U.S. Nuclear Regulatory Commission, "Fort Calhoun Station – NRC Follow-up Inspection – Inspection Report 05000285/201007; Preliminary Substantial Finding," NRC Inspection Report 05000285/20010007, July 15, 2010.

[88] Dictionary.com, "The_power_of_generalizing_ideas_of_drawing_comprehensive," in *Columbia World of Quotations*. Source location: Columbia University Press, 1996. <http://quotes.dictionary.com/The_power_of_generalizing_ideas_of_drawing_comprehensive>. Available: http://dictionary.reference.com. Accessed: April 27, 2012.

Appendix H: Example checklist of NPP modes

1. On Power
   1.1. Full allowable power
   1.2. Reduced power (including zero power)
   1.3. Raising power or starting up
   1.4. Reducing power
2. Hot Shutdown (reactor sub-critical)
   2.1. Hot standby (coolant at normal operating temperature)
   2.2. Hot shutdown (coolant below normal operating temperature)
3. Cold Shutdown (reactor subcritical and coolant temperature < 93 °C)
   3.1. Cold shutdown with closed reactor vessel
   3.2. Refueling or open vessel (for maintenance)
      3.2.1. Refueling or open vessel – all or some fuel inside the core
      3.2.2. Refueling or open vessel – all fuel outside the core
   3.3. Mid-loop operation (PWR)
4. Construction
5. Preoperational
6. Startup test
7. Commissioning
8. Testing or maintenance being performed
   8.1. Setpoint adjustment
   8.2. Instrument calibration
   8.3. Change (switching) of calibration parameters (in [21] CP 2.1.3.2.5)
9. Decommissioning

Appendix I: EVALUATION OF TIMING ANALYSIS

Author: Professor Dr. John Stankovic, University of Virginia
http://www.cs.virginia.edu/people/faculty/stankovic.html

Integrative editing by Sushil Birla, Senior Technical Advisor for I&C, U.S. Nuclea Regulatory Commission

This appendix summarizes the state of the art in timing analysis. Timing analysis is used in design to evaluate its suitability to support timing and related constraints. Timing is re-analyzed to confirm satisfaction of these constraints after implementation using actual execution times and delays.

A design description should include the approach being taken to guarantee timing behavior with accompanying timing schedules and resource assignments that *logically* guarantee timing. An evaluator can expect to see different approaches. However, it is very unlikely that there exists an exact case study or exact match between the principles described below and the system under evaluation. It will be necessary for the evaluator to apply significant knowledge and expertise in real-time theory and practice.

In performing timing analysis there are at least four overarching approaches that could be presented by the developer. First (Sections I.1 and I.2) is a complete and explicit layout of all tasks on time lines that represent a deterministic execution time for everything and in such a manner as to meet all timing, ordering, and resource constraints. This would include identifying the processing elements (CPUs, FPGAs, etc.), the assignment of tasks to each processing element, and message slots on busses and their purpose. Another proposed approach might be the use of fixed priority scheduling. This means that the operating system on each processing element runs tasks according to fixed priorities as assigned by the developers to guarantee timing. This approach should be supported by fixed priority mathematical analysis (Section I.3.1). Another approach may be to use dynamic priorities and apply its associated analysis (Section I.3.2). This approach is less deterministic, but has advantages in many situations and can be used as an off-line analysis to guarantee timing. A fourth approach is use of FPGAs (Section I.4). In all the design approaches, realistic, but estimated times should be identified. Accounting for redundancy and fault tolerance techniques in the design must be included. Consider each of these in more detail.

I.1 Timing analysis by hand

The developer, using a by-hand approach, may present a set of time lines with all tasks assigned deterministically. How they created these time lines (possibly by hand) may not be known and is generally very complex. For the evaluator, once the deterministic time lines are given, it is much simpler to check (one by one) if the set of assignments and time lines meets all the timing, ordering, and resource constraints. This approach is sometimes used for small and simple subsystems. It is not recommended for complex designs since any change at all results in a complete re-creation of the timelines and allocations which is error-prone and costly.

I.2 Timing analysis by a program

In this approach a developer may create the deterministic time lines and assignments using some algorithm or heuristics implemented as a computer program. The evaluator would analyze the resultant schedules as in Section I.1. This approach is more desirable than in Section I.1, since changes can be more easily handled rather than having to recreate schedules and time lines by hand. Cyclic schedulers and time triggered approaches [1] are examples of this approach.

I.3 Mathematical Analysis of timing

Many analysis techniques might be applied to the design. Two of the most common are fixed and dynamic priorities. These both assume that there is an underlying operating system (OS) that executes tasks based on priority.

I.3.1 Mathematical Analysis of timing with fixed priorities

Rate Monotonic (RM) analysis [2] is a set of techniques to assign fixed priorities and perform an associated timing guarantee analysis. It focuses on periodic tasks, but can be extended to address both periodic and aperiodic tasks. RM analysis can incorporate the complexities discussed below in section I.1. As an example, for a large number of periodic tasks if the sum of the cpu utilizations of these tasks is below 69% then it is guaranteed that all deadlines will be met. This is true even though there are preemptions. RM analysis assumes that deadlines equal periods. If deadlines are less than periods then a different set of analysis is required, called Deadline Monotonic (DM) [2]. RM has been used successfully in some avionics systems and control systems in automobiles.

I.3.2 Mathematical Analysis of timing with dynamic priorities

Dynamic priorities normally refer to the OS scheduler choosing the next task to execute based on current task priorities which can change at runtime. These solutions are usually based on the earliest deadline first (EDF) algorithm. However, if all tasks and their requirements are known at design and implementation time, then EDF and its analysis [3] can be applied off-line, and timing guarantees are possible. In this case the results are very similar to the fixed priority approach except the OS is running an EDF scheduler instead of a fixed priority scheduler. An evaluator may also see EDF as a basis for the By-a-Program approach mentioned above.

I.4 FPGAs

Various functions in the system may be implemented in hardware (today typically via an FPGA) [5]. Then execution speed of the function can be greater than on a CPU. Functions implemented on a FPGA can be considered tasks in the overall timing analysis and subject to the analysis techniques[155] described in this appendix. Of course, issues such as I/O, ordering, synchronization etc. must all be considered.

I.5 Practical considerations in applying mathematical analysis

Basic scheduling theory is often presented with many simplifying assumptions. Fortunately, many practical issues can be addressed with extensions to the basic theory for analysis.

I.5.1 Interrupts

Sometimes interrupts may be necessary. By careful design it is possible to limit the maximum number of interrupts. The time it takes to handle each interrupt can be bounded. Consequently, the basic timing analysis can account for the worst case delays for task executions due to interrupts. See Ch. 5 in [2].

I.5.2 Resources

[155] National Instruments (LabView development system together with the Real-Time Module and and FPGA module) is an example of a source of tools currently available for use in common practice.

Tasks often require resources beyond the cpu, e.g., access to a data structure or bus. Tasks can contend for these resources. In addition, to guaranteeing no deadlock it is necessary to determine the worst case blocking delay for any exclusively shared resource. In RM analysis this is handled by the priority ceiling version of RM - see pages 5-47 to 5-60 in [2]. For EDF see Ch 7 in [3].

I.5.3 Ordering

In many systems a set of tasks must execute in a fixed order. For example, the sensor must first sample, AD conversion must execute, the result then sent to a processor, a task executes to process the data, this result is then converted to an actuator control, and possibly also sent to a display. Classical scheduling theory has many results for job shop scheduling in this area. Ordering constraints can also be imposed on task sets when using cyclic, time triggered, RM or EDF based approaches. See pages 3-10 to 3-11 in [2] and Ch 7 in [3].

I.5.4 I/O

Any inputs for tasks must be ready when an instance of a task is “released” for execution. This is normally analyzed as precedence constraints. If the task produces an output it must be made clear when that output happens, e.g., only at the end of execution of the tasks or possibly at any point within the execution of the task. Controlling jitter is often necessary for I/O. See Ch. 6 in [2].

I.5.5 Distributed Systems

Communication between distributed parts of a system introduces delays. Such delays can be deterministic if bus slots are defined and allocated. Redundant slots can be allocated for fault tolerance. The time triggered approach is a well-known way to do this [1]. These communications delays can also be addressed by RM (Ch. 6 in [2]).

I.6 Caveats and Things to Watch Out For

Timing design and analysis is very difficult and fraught with hazards. A slight change in assumptions can make a major difference in the accuracy of the analysis. Following are some examples of common misunderstandings.

I.6.1 Task semantics

Most periodic task analysis assumes that the semantics of a task period means that a task executes once per period P. This does NOT guarantee a minimum or a fixed time between two instances of a periodic task. For example, with this semantics two executions of a task could run back-to-back without any time interval between them.

I.6.2 Non-determinism introduced by hardware

Worst case execution times must be determined for tasks. This is difficult to determine and often just measured which is not recommended. Measurements can be way off if non-deterministic features on hardware, such as caching, branch prediction, virtual memory, or multi-core contention, are involved.

I.6.3 The overhead of the OS

Logical analysis may not account for the time it takes to select and switch between tasks. This would be incorrect. See p. 392-395 in [4].

I.6.4 Richard's Anomalies

Scheduling can lead to hazardous conditions subtly. For example, if a set of time lines is analyzed as correct and then the developer decides to use faster processors (maybe with idea to give more slack time thereby increasing a safety margin), then the previous schedules which worked (i.e., all deadlines met) may now miss deadlines even though individual tasks are executing faster. There are 4 variations of these anomalies (pages 42-51 in [4]).

I.6.5 Overloads

Many hard real-time systems assume that all timing is guaranteed so there is no such thing as an overload. Safety margins can be built into task execution times and resource requirements to make overload even less likely. However, understanding the consequences of an overload, even if not expected, is important. Will the system fail safe? Could there be a catastrophic cascade of deadline misses due to the overload? See Chapter 9 in [4].

I.7 Integrating timing analysis in engineering

See [6] for an approach to integrate timing analysis in model-based engineering.

References for Appendix I

[1] H. Kopetz, Real-Time Systems, Second Edition, Springer, 2011.

[2] M. Klein, T. Ralya, B. Pollak, R. Obenza, and M. Gonzales Harbour, A Practictioner's Handbook for Real-Time Analysis, Kluwer, 1993. (second printing 1994)

[3] J. Stankovic, M. Spuri, K. Ramamritham, and G. Buttazzo, Deadline Scheduling for Real-Time Systems, EDF and Related Algorithms, Kluwer, 1998.

[4] G. Buttazzo , Hard Real-Time Computing Systems, Predictable Scheduling Algorithms and Applications, Springer, Third Edition, 2011.

[5] National Instruments LabVIEW FPGA Module. URL: http://sine.ni.com/nips/cds/view/p/lang/en/nid/11834

[6] P. Feiler and D. Gluch, Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language, Published Sep 25, 2012 by Addison-Wesley Professional. Part of the SEI Series in Software Engineering series.

Appendix J: ASSUMPTIONS

Author: Professor Dr. John Stankovic, University of Virginia
http://www.cs.virginia.edu/people/faculty/stankovic.html

Integrative editing by Sushil Birla, Senior Technical Advisor for I&C, U.S. Nuclea Regulatory Commission

In reasoning that is part of safety analysis, an assumption is a premise that is not yet validated. Explicit assumptions are documented. Implicit assumptions are not documented, because they are not known or understood or were lost over time. Assumptions, especially implicit assumptions, that turn out to be invalid (not true) are the root cause of many system failures and a contributor to hazards in many other cases. An initially valid assumption may become invalid over time. It is also common that combinations of assumptions may cause failure or contribute to hazards. For example, a component (hardware or software) may get reused without full awareness and consideration of assumptions that invalidate its fitness for reuse in a different context. Assumptions occur in every phase of the system development lifecycle (e.g.: requirements; design and analysis; implementation; testing). Overall, it is necessary to document, manage and assess the impact of assumptions throughout the life cycle, particularly if some critical property of the system, such as SAFETY, has to be assured.

Assumptions often affect timing analysis - see Appendix I. Assumptions also affect dependencies - see Appendix K.

J.1 Systematized consideration of assumptions – state of the art

There is a lack of accepted approaches towards systematic assumption declaration, management, and assessment. Statement of assumptions may be classified in three ways:

1. Formal-like languages: For example, in AADL [1] an assumption can be stated with an ***assumes*** keyword and some written in predicate or temporal logic. Then automatic assumption matching checks can be run.
2. Semi-Formal: For example, in XML, an assumption may be may be categorized by type (e.g., see Table 27) and incorporated into an *assumption management system* [2], as shown in Figure 14.
3. Informal: Used mostly in current practice, an assumption is stated in a natural language such as English. Because it is subject to misinterpretation, which can contribute to hazards, it is not adequate and not recommended for use in engineering a very critical system.

Assumptions can also be categorized as static and dynamic assumptions and indicate a level of criticality. These notions should be part of the assumption descriptions.

```
<assumption>
    <type>
                Control
    </type>
            <description>
        Statement of control assumption in a previously declared language.
            </description>
  </assumption>
```

Figure 14: Example of semi-formal statement of an assumption

Table 27 includes the different types of assumptions which could be stated in XML, with brief associated examples.

Table 27: Different types of assumptions which could be stated in XML

Type of Assumption	Example of an informal statement of an assumption
Management	Person X is responsible for a particular task.
Environment (of system)	Back-up power is available 24x7.
Software Component Design (Decisions)	Minimum amount of data required for a component to make a decision is <…>.
System Software	Background processing runs at infrequently scheduled times.
Hardware	Caches are not to be used.
Timing	Some declared minimum time must elapse between two consecutive executions of a task.
Control	Only one module must control a particular actuator.
Data	Data set X must be replicated at physically distinct memories.
Semantics of Application	Property X exists for a given component when executed, e.g., the accuracy of a signal processing module when assessing critical condition of the plant.
Faults	A particular fault will not occur more than x times in interval y.
Security	Communication X is encrypted.

When an assumption is stated in this form, a management system can analyze it for potential problems (e.g., contributory hazards) and updates can occur over time. For example, the analysis may find that across the entire set of assumptions there are two or more assumptions that cannot simultaneously be true. It is also possible to match assumptions among composed components. Some software development kits, such as Eclipse [3], integrate environment, assumptions, architecture and source code in the same tool.

A complex system may entail an enormous number of assumptions of all types (Table 27) and for various purposes (Table 28).

Table 28: Examples of assumptions for different purposes

Context of Assumption	Example of assumption
Timing	All worst case execution times are known.
Timing	All tasks always meet their deadlines. (What is the impact of a task missing its deadline?)
Timing	There is enough memory assigned to each task.
Timing	No hardware will be changed; etc.
Fault tolerance	On power failure a battery backup is available and it is functional
Fault tolerance	More than “n” simultaneous failures do not occur.
Security	A particular module will not be attacked.
Security	An encryption key won’t be compromised
Control	Only one module controls a particular actuator.
Control	Data sent to control algorithm is correct and in-time.

J.2 Monitoring an assumption at run time

Since underlying assumptions have been the cause of many failures and can contribute to hazards, assumption-aware work products of engineering are valuable – indeed, necessary, in complex critical systems (e.g., for which the SAFETY property has to be assured). If an assumption can change over time, runtime monitoring for such change may be considered.

Does the presented design have an assumption that can change over time? If so, does the design include run time monitoring of the change in the assumption?

J.3 Statement of assumptions within code

Sometimes, assumptions are also written into source code (with a keyword such as *assumes*), so that source code can be scanned by programs to collect and analyze all the assumptions. This technique often deteriorates over time as code is updated and assumptions are not.

J.4 Statement of assumptions within models

Assumptions can also be added to graphic representations of work products, using tools based on languages such as SysML [4]. This tends to be imprecise and difficult to maintain. Academic tools such as Ptolemy have some support for specifying assumptions [5].

J.5 References for Appendix J

[1] P. Feiler, D. Cluch, J. Hudak, The Architecture, Analysis and Design Language (AADL), An Introduction, CMU/SEI-2006-TN-011.

[2] G. Lewis, T. Mahatham, and L. Wrage, Assumptions Management in Software Development, Technical Note, CMU/SEI-2004-TN-021, August 2004.

[3] Eclipse http://www.eclipse.org/home/index.php

[4] SysML: http://www.sysml.org/

[5] Ptolemy: http://ptolemy.eecs.berkeley.edu/

Appendix K: DEPENDENCY

Authors:

Prof. John Stankovic, University of Virginia http://www.cs.virginia.edu/people/faculty/stankovic.html

Prof. Manfred Broy, Technische Universität München http://www4.in.tum.de/~broy/

Prof. John McDermid, University of York http://www-users.cs.york.ac.uk/~jam/

Integrative editing by Sushil Birla, Senior Technical Advisor for I&C, U.S. Nuclea Regulatory Commission

K.1 Purpose and Scope

This appendix explains the term, dependency, as used in Section 2.7.6.

In software it is often noted that if module $\mathcal{A}$ *uses* module $\mathcal{B}$, then module $\mathcal{A}$ depends on module $\mathcal{B}$. However, dependencies are much more complicated than a simple *uses* relation. This appendix provides a comprehensive understanding of these complications.

A dependency between two or more system elements may exist or occur through their structure, their behaviors, or their values, in the form of some cause-effect relationship.

There exist a number of dependencies within developed systems, between their elements and their constituents as well as in their descriptions as included in their work products [1].

K.2 Safety significance of dependency

A safety system in an NPP is an independent layer of defense. An independent layer of defense protects against the unknowns and uncertainties in the other layers of defense. An obscure dependency can undermine the intended defense strategy.

Dependencies on common sources of defects or deficiencies can render homogeneous redundancy ineffective, because the same defect can repeat in each redundant element; for example:

- Defect or deficiency[156] in a requirement.
- Defect or deficiency in implementation of the application software.
- Defect or deficiency in implementation or configuration of the system software.

Dependencies can propagate the effect of a deficiency to independent and functionally different units; consider the following cases:

- Dependency on common internal information; for example:
  - Year 2000 “bug.”
  - Count of cycles since the last reset.
- Dependency on conditions, external to the units; for example:
  - Usage of resources dependent upon process transients.

Section 3.4.2 Item 3 refers to the concern of compromise of redundancy through a dependency.

The effect of these dependencies should be analyzed to prove that the safety function is not degraded.

[156] Issue: If requirements are deficient, the terms “failure” and “defect” are not applicable; the CCF notion, applied to a specified system, does not serve well; failure analysis and defect analysis do not serve as adequate hazard analysis.

K.3 Types of Dependency

Any factor on which an identified hazard depends (or by which it is influenced) is a contributory hazard. Dependency may be through many kinds of coupling[157]; for example:

1. Function.
2. Control flow.
3. Data; information.
4. Sharing or constraint of resources.
   4.1. Explicit preference-order.
5. Conflicting goals or losses of concern.
6. States or conditions in the environment.
   6.1. Controlled processes.
   6.2. Supporting physical processes.
7. Fault
8. Constraints
9. Assumptions
10. Concept.
11. Some unintended, unrecognized form of coupling.

K.4 Examples of dependencies

Dependencies exist within and across hardware and software components and also result from the interaction with the physical world. To organize the ideas of dependencies we first list and give a few examples of those dependencies that arise from the hardware and the physical world.

1. Sensors: software signal processing and decision making algorithms are dependent on the properties of sensors such as range, accuracy, repeatability, sensitivity, resolution, overshoot, drift, and power as well as the numbers and placement (location) of the sensors.
2. Actuators: power needed to run the actuator; accuracy of applying command signal
3. CPU/Memories: speed of cpu; implementation features such as caches, branch prediction; size of memory; type and location of memories on busses; power requirements
4. FPGAs: speed; power; timing; availability of inputs
5. Busses: communication between distributed devices/software depend on the bus speed and access protocols; may also depend on a hierarchy of busses.
6. I/O devices: speeds; power; location; read and write techniques
7. Physical properties: sizes of sensors, actuators, computing; I/O; temperatures produced by devices; reliability of devices; fault models; will system degrade over time without renewing/maintenance (a form of entropy)

[157] In addition to the factors directly in the causal paths, hazards can also be contributed from side effects such as interferences across activities and resources.

8. Time: guaranteeing deadlines depends on the time requirements of real world phenomena, the speed of hardware, the software processing required, scheduling algorithms; accumulated delays
9. Location: where sensors, actuators, displays are placed
10. Environmental: external conditions such as earthquakes, hurricanes, power outages; humidity; fire
11. Control: accuracy of models upon which control algorithms were created; availability or max delay of inputs to controller
12. Chain of events: collection or order of sets of events
13. Humans: reaction time; awareness; expertise

Examples of dependencies that arise primarily in software include the following:

1. Numbers and types of parameters: This is straightforward to check and often given in Interface Definition Languages (IDLs).
2. Uses relationship: A call graph (usually automatically generated) can identify simple uses relationships.
3. Runtime environment: The OS, its version, and particular settings (configurations) and algorithms being used constitute the runtime environment. It is necessary to ensure that there are not unexpected modules being run, e.g., for system monitoring or periodic cleanup modules unless required and accounted for.
4. Resources: amount of cpu time, memory, bus bandwidth
5. Name: components assumed named consistently
6. Data: location, synchronization, availability, redundancy
7. Ordering: some sets of components must run in a strict or partial order
8. Race Conditions: if components can execute in different orders the result may be a race condition.

The following examples demonstrate how tight specifications, assumptions, and constraints interrelate logically and may lead to implicit dependencies that can be discovered by analysis of explicitly documented dependencies.

K.4.1 Example of a data dependency

For instance, two state attributes, $\mathcal{A}$ and $\mathcal{B}$, for data values in a system are in a dependency if given the value of $\mathcal{A}$, the value of $\mathcal{B}$ is affected by the value of $\mathcal{A}$ (e.g.: fixed to a specific value; bounded within a specific range).

K.4.2 Example of a timing dependency

Other examples are timings of events or causal dependencies between events such as shown in the following simple example

- event A: “temperature of water gets too high while valve is closed”;
- event B: “valve opens”;

- dependency in the system: “whenever event A happens then event B happens within x milliseconds”.

K.4.3 Example of a dependency on a hardware function

A function or information in software can be dependent on a function implemented in hardware. Example:

“sensor 1 data available” **depends_on** “power supply X failure”

“sensor 2 data available” **depends_on** “power supply X failure”

which indicates a common cause failure. Such a dependency is different from direct dependency.

A common cause dependency between events A and B can be denoted as follows:

common_cause A, B

if there is an event C where the condition

A **depends_on** C

and

B **depends_on** C

holds.

K.4.4 Example of a resource dependency

These different types of dependencies may interact. For instance, a resource dependency may cause a functional dependency. If there are two functions, $\mathcal{A}$ and $\mathcal{B}$, that as intended to be independent but use the same resources, unintentionally become dependent. If function $\mathcal{A}$ may compromise the shared resource in a certain situation such that the function $\mathcal{B}$ is no longer available, this is an unwanted (and unspecified) dependency from $\mathcal{A}$ to $\mathcal{B}$.This example illustrates that the hazard analysis of systems should consider the logical relationships between dependencies and should consider rules to deduce further dependencies from explicitly documented ones.

K.4.5 Dependency through assumptions and constraints

There are constraints on interactions that cause dependencies:

- Assumptions about the environment: properties of the environment that are represented as assumptions (example: “The water temperature cannot change by more than 10 degrees with 10 milliseconds”).
- Properties of system elements: interact with assumptions (example: “Whenever the temperature changes by more than 1 degree the sensor issues a signal”).
- Constraints on interactions (example: “There is a delay of at least 1 millisecond between two signals issued by the temperature sensor”).

Assumptions are often not given to the developer as part of the specification and are not direct relationships between components of the system. Note that the overall system depends on assumptions being valid so there are dependencies related to assumptions, but they are treated separately (see Appendix J: ASSUMPTIONS).

K.4.6 Example of logical dependency between logical entities

Let us consider examples of system properties expressed by logical entities:

(*P1*) "The temperature changes within 1 millisecond by less than 1 degree."

(*P2*) "The temperature sensor updates the variable that stores the measured temperature every 10 milliseconds."

(*P3*) "The variable that stores the measured temperature holds a value that deviates at most by 10 degrees from the actual temperature."

These logical entities may be contained in different work products or in one work product at different positions.

(*P3*) expresses a system dependency.

(*P3*) is a logical consequence of (*P1*) and (*P2*). This is an example of a dependency between logical entities.

If the property "The water is too hot" is a hazard (or a contributing hazard) and if its mitigation depends on the preciseness of the stored measured temperature, then the dependency "(*P3*) is a logical consequence of (*P1*) and (*P2*)" is of relevance for the hazard analysis. If (*P1*) or (*P2*) are changed, then the conclusion of the hazard analysis may no longer be valid.

Specific logical dependencies may relate logical entities formulated at different levels of abstraction. Assume that a sensor sends an alarm signal *S1* if the water temperature gets too hot. Then the dependency between event "signal *S1* sent" and the event "water temperature too hot" is only understandable by the additional information "signal *S1* indicates water too hot". This way we get a relationship between the technical information "signal *S1* sent" and the domain specific event "water temperature too hot".

For dependencies between system properties the dependency model basically captures logical dependencies between logical statements (in terms of logical entities). The network of logical dependencies is basically addressing logical implication. Although logical implication seems to be a rather straightforward concept, as well known given a number of logical propositions (documented by logical entities), their implication relationships can be very sophisticated by combining them in applying deduction rules leading to proof trees, which represent sub-networks of the networks of logical dependencies (see [1]).

K.5 Dependencies can network

For a system of the kind in RIL-1101 focus, dependencies are not simple chains or trees, but a network (also known as directed graph or digraph [64]); for example:

- The same factor may recur in many places in the network (i.e., common causes).

- There are feedback paths; the dependency structure is a directed cyclic graph. It is a well-known generic control structure, for which well-known analysis techniques exist. It can be applied to a safety-related system in its concept phase (Section 3.4) or to its element (Sections 3.7; 3.8-3.9). It can also be applied to the technical processes (Section 3.3), for developing a safety related system or its element. It can also be applied to the organizational processes (Section 3.2) that influence the development processes.

K.6 Dependencies can propagate through faults

Also for faults in systems there exist many dependencies. Hazard analysis should include the analysis of the dependencies across faults to find whether a fault can propagate to degrade a safety function. This requires (see [3][158]) a fault propagation specification and component fault behavior specification, an explicit specification of fault types propagated, and an explicit specification of system fault states.

K.7 Unrecognized dependency

Missing, wrong, unwanted, or misunderstood dependencies may contribute to a hazard. If $\mathcal{A}$ can have an unwanted effect on $\mathcal{B}$, then $\mathcal{B}$ is in some sense dependent on $\mathcal{A}$. In other words, $\mathcal{B}$ is not independent of $\mathcal{A}$. Dependence of this type motivated RIL-1101, in which it is characterized as Interference. Furthermore, in such cases (of unwanted interactions), the effect on $\mathcal{B}$ may not be determinable. For example, consider

- effect of resource sharing
- effect of a memory leak.

There are so many sources of unwanted dependencies that it is easy[159] to miss one. As soon as one is discovered or suspected, it should be documented. Then, known methods can be applied to perform the analysis.

Unrecognized dependencies are defects in hazard analysis and may lead to degradation of a safety function.

For complicated dependencies many observations are needed to uncover dependency [2].

K.8 Expressing dependencies

System dependencies are general relations between

- system functions
- system elements
- platform (infrastructural) services
- system events, messages, and signals
- system data
- system states

[158] This reference uses the term “error” which is mapped into the term “fault” in RIL-1101.

[159] In current practice

- system timing

This documentation can be made very explicit (example: proposition *P1*: “event A leads to event B”; proposition *P2*: “event B leads to event C”) or implicit where a dependency can be concluded from explicit stated dependencies (example: from the two propositions *P1* and *P2* we can conclude proposition *P3*: “event A leads to event C”). If all three propositions *P1*, *P2*, and *P3* are explicitly included in work products in logical entities, say E1, E2, and E3 resp., then we get an instance of dependencies between logical entities of work products. The contents of E1 and E2 imply proposition *P3* being part of the content of E3.

The following predicate expresses dependencies in a formalised way for events A and B in a system[160]:

A **depends_on** B

This proposition expresses that there is some causal relationship between A and B. Actually there are many instances for such a relationship:

- A cannot happen before B has happened; example: consider a system which is supposed to raise an alarm (event A) as soon as the pressure in a tank gets too high and where there is a sensor that measures the pressure and sends the values to the alarm manager (event B).
- A is guaranteed to happen if B has happened; example: consider a system which is supposed to raise an alarm (event A) as soon as a the pressure in a tank gets too high and where there is a sensor that measures the pressure and sends the values to the alarm manager; then the “incorrect pressure too high data measured at sensor” (event A) leads to an incorrect alarm (event A).
- A cannot happen if B has happened; example: consider a system which is supposed to raise an alarm (event A) as soon as a the pressure in a tank gets too high and where there is a sensor that measures the pressure and sends the values to the alarm manager over a communication line; assume that the energy supply for the communication line can be interrupted (event B).

Note that the proposition

A **depends_on** B

does not require that in every behavior the event A may interfere with B; it means that there is some instance of behavior where A does interfere with B.

Note furthermore that the relationship

A **depends_on** B

Is not symmetric, in general, and even not transitive. The same holds for its negation

A **independent_of** B

[160] This is an example of a formal predicate on dependecies between events. Similar relationships can be introduced between data attributes in states or, more generally, rules for dependencies in data and control flow and how their dependencies relate.

The missing transitivity of the independence relation makes it very difficult to reason about independence and freedom from interference.

The examples show that dependencies between system constituents lead to dependencies between logical entities of work products. Since the content of logical entities of work products can be understood as logical propositions and predicates, these dependencies can be treated as logical relations between propositions or predicates.

System dependencies can be reflected in system models. The models should contain enough information to understand dependencies and propagation paths for contributory hazards (see Table 22 Note 1. Appendix C.6 suggests how a dependency model can help HA.

A model captures and describes certain classes of dependencies (such as process dependencies) including rules to derive dependencies and to analyze their effects. This does not imply that a separate model is needed exclusively for this purpose. A separate model could lead to inconsistencies with the primary engineering model. For dealing with dependencies within the work product, the primary engineering model of the (work) product should suffice: model of requirements; model of architecture; model of detailed design; source code could also serve as a “model” of the executable. These models should be expressive enough to capture all kinds of dependencies.

For dependencies within the development process, the primary engineering model of the process should suffice. In other words, all factors affecting the product (of the process) should be identified in the process model.

Semantics of the relationships should be explicit.

K.9 Deriving dependencies

Note the difference between an implicit dependency, which is not documented explicitly, but can be deduced by combination from explicitly documented dependencies, and a dependency that is not identified at all - thus, not discoverable through analysis.

The system behavior can be deduced from the architecture and the specification of the interface behavior of its elements, when rules of composition and refinement are followed (see Appendix D: REFINEMENT). Similarly, system behavior can also be deduced from some fault condition in an element of the system, if the architecture includes the relationships that affect fault propagation [4].

Thus, a well-specified architecture is essential for dependency analysis (see [3][4]).

K.10 Avoiding unwanted dependency

Careful explicit specification of constraints and system properties and subsequent analysis make hidden dependencies explicit and help to avoid unwanted dependencies and to reason about dependencies in hazard analysis.

K.11 Languages available for modeling dependencies

Examples of means that have been used to model[161] dependencies include the following: Call graphs, IDLs [5], data flow diagrams, and design languages (graphical or not) such as AADL [6] and SysML [8]. AADL, with extensions and supporting tools, is in use as a research platform in many countries, with ongoing extension activities to support safety evaluation [3].

For Want of a Nail

For want of a nail the shoe was lost.
For want of a shoe the horse was lost.
For want of a horse the rider was lost.
For want of a rider the message was lost.
For want of a message the battle was lost.
For want of a battle the kingdom was lost.
And all for the want of a horseshoe nail.

K.12 References for Appendix K

[1] M. Broy: A Logical Approach to Systems Engineering Artifacts and Traceability: From Requirements to Functional and Architectural Views. In: M. Broy, D. Peled, G. Kalus (eds): Engineering Dependable Software Systems, IOS Press 2013, P. 1-48.

[2] J. L. Pfaltz: Logical implication and causal dependency. Conceptual structures: inspiration and application, volume Springer Verlag LNAI, 4068, 145-157. (2006).

[3] J. Delange, P. Feiler: Supporting Safety Evaluation Process using AADL. Layered Assurance Workshop '2013, New Orleans, USA

[4] P. Feiler, A. Rugina: Dependability Modeling with the Architecture Analysis & Design Language (AADL). CMU/SEI-2007-TN-043, Technical report, Carnegie Mellon Software Engineering Institute, July 2007

[5] http://www.omg.org/gettingstarted/omg_idl.htm

[6] P. Feiler, D. Cluch, J. Hudak, The Architecture, Analysis and Design Language (AADL), An Introduction, CMU/SEI-2006-TN-011.

[7] S. Si Albir, UML in a Nutshell, O'Reilly, 1998.

[8] http://www.sysml.org/

[161] These are not necessarily complete and are only as good as the information recorded in them.