RESEARCH INFORMATION LETTER 1101:

Technical Basis to Review Hazard Analysis of Digital Safety Systems

EXECUTIVE SUMMARY

The Office of Nuclear Regulatory Research (RES) prepared RIL-1101 in response to an Office of New Reactors (NRO) user need request, dated December 8, 2011. NRO requested technical basis for the regulatory review of an applicant's hazard analysis (HA) and corresponding acceptance criteria relevant to digital instrumentation and control (DI&C) safety systems of nuclear power plants (NPPs). The requested information supports improvements to the regulatory guidance for evaluation of an applicant's HA.

The technical basis provided in RIL-1101 focuses on evaluation of an applicant's HA—rather than performing HA—while also addressing challenges that NRO has encountered during its licensing reviews. Many of these challenges come from potential hazards that are rooted in systemic causes, such as inadequacies in engineering organizations, processes or methods. RIL-1101 refers to systemic causes as contributory hazards. RIL-1101 identifies systemic causes of DI&C safety system developments that may contribute to hazards. When a systemic cause can adversely affect an NPP DI&C safety system, RIL-1101 considers it a contributory hazard.

RIL-1101 provides the U.S. Nuclear Regulatory Commission's (NRC's) licensing staff technical basis to create regulatory guidance for evaluation of an applicant's HA for DI&C systems. An applicant's HA for a design certification or license amendment, which involves a DI&C safety system, establishes design bases of the plant and for its digital safety systems. Where RIL-1101 identifies contributory hazards relevant to DI&C safety systems, RIL-1101 also identifies conditions to address them and reduce the hazard space. These conditions to reduce the hazard space represent technical basis for potential acceptance criteria for regulatory reviews of future new and advanced reactor applications.

Hazards are the potential for harm (e.g., radiological consequences leading to disease, loss of life, damage to the environment, etc.). To prevent these hazards, nuclear power plant I&C systems maintain plant processes within acceptable performance limits by making reliable and accurate measurements that lead to reliable, accurate, and timely control actions. Using redundant, independent, electrically-isolated, and physically-separated components, I&C safety systems sense plant conditions and actuate controls before a limiting safety setting is exceeded to preserve fuel and reactor vessel integrity.

RIL-1101 identifies examples of hazards and factors that contribute to hazards by degrading the safety function of a DI&C system. DI&C systems differ from their analog and mechanical counterparts. Rapid changes in digital technology prevent accumulation of the kind of operating history that applies to analog and mechanical systems. Many unsafe behaviors of digital I&C systems do not relate to physical principles like those used to evaluate the safety and reliability of analog and mechanical systems. Instead, malfunctions of DI&C systems more often arise from systemic causes associated with characteristics of their design and development. These characteristics can also make verification of DI&C systems more difficult when compared to

analog or mechanical systems. Furthermore, analog and mechanical systems have interconnections, dependencies, and interactions that are readily apparent to a reviewer as wires and pipes. In contrast, DI&C systems have interconnections, dependencies and interactions that are less obvious. When unrecognized these less obvious attributes can degrade the safety benefit presumed to exist through redundant, independent, electrically-isolated, and physically-separated safety components. RIL-1101 addresses each of the considerations, which are unique to DI&C systems.

Adopters of the hazard-analysis approach in RIL-1101 can apply it to an early-stage functional concept, and iterate the approach on the successive work products, as the development progresses. When applying the hazard-analysis approach in RIL-1101, the resulting design criteria and design bases would include constraints that avoid conditions that contribute to hazards. Early Identification of these avoidable contributory hazards and constraints to eliminate them drive downstream engineering to prevent later problems. The prevention of problems earlier in the lifecycle improves lifecycle economics while increasing safety.

Contents

Figures

Tables

1 INTRODUCTION

This research information letter (RIL) provides the U.S. Nuclear Regulatory Commission's (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, combined license, or a license amendment. Section 1.5 provides a brief background on HA, supported with elaboration in Appendix C. Section 1.6 states the purpose and intended audience.

1.1 Regulatory basis

Hazard analysis of a digital safety system could address clauses 4.8 and 5.6 in IEEE STANDARD 603-1991 [1], incorporated by reference in 10 CFR 50.55a(h)(3) [2]. Hazard analysis of a digital safety system could contribute to the analysis aspect of Title 10, "Energy," of the *Code of Federal Regulations* (10 CFR) 50.34(a)(3) [3] and 10 CFR 52.47(a)(2) [4].

1.2 Work authorization

The RIL has been prepared in response to a non-publicly available user need request, NRC Agencywide Documents Access and Management System (ADAMS) Accession No. ML11313A214, 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). 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.

1.3 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 standalone technical reference for reviewing HA of digital safety systems in NPPs. Section 1.7 elaborates on the scope. Section 1.8 explains the organization of the RIL.

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

- Rapid changes in the nature of systems and the underlying technologies (H-OTproc-7) and
- Increasing interconnectivity (Sections 2.4.1 and 2.4.2), resulting in
- Less accumulated experience for hazard analysis of each kind of 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, and analyzability) associated with safety functions.
- Incorrect flowdown into constraints on the architecture of the system and then the architecture of the software or other forms of logic.
- Inadequate flowdown to identify requirements and constraints on technical processes, supporting processes, and organizational processes.
- Declining supply and replenishment of requisite competence (Section 2.1; H-0-2).
- Longer supply chains with weaker communication links (Section 2.1.2 H-0-9 item 2).
- Inadequate quality of cross-organizational cross-disciplinary communications, etc. (Section 2.2; H-culture-9).

1.4 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-and-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 form 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; Appendix A of NRO's mPower design-specific review standard (DSRS) [5] is an example.

1.5 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 assigned 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 hazards[4], as well as the requirements and constraints to eliminate, prevent, or otherwise control those hazards.

HA is a subset of safety analysis and its evaluation is a subset of safety evaluation; the relationship is explained in Section 1.7.8.

[1] In common practice, these are treated as "nonfunctional" requirements.

[2] Hazard space is defined as "all of the possible combinations of specific conditions that are relevant to a scenario that could lead to the degradation of a safety function."

[3] This implies the use of systematic and repeatable methods for performing HA.

[4] Which include causal factors.

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. 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.

1.6 Purpose and intended audience

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

Because the NRC has not previously provided 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 presents a learning opportunity from which NRC expects to identify needs for future improvements in its review guidance, regulatory guidance, and the underlying technical basis (i.e., the successors to this document).

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 might also be useful to stakeholders outside the NRC.

1.7 Scope

The RIL is a response to NRO’s user need request for supporting a specific project. However, the content is sufficiently generic to be used to generate a successor for broader application after RES learns from NRO’s first experience (Section 1.7.1). Content has been selected to support evaluation rather than performance of HA (Section 1.7.2) for NPP digital safety systems (Sections 1.7.3 through 1.7.5). Content is focused on hazards contributed through systemic causes, especially inadequacies in engineering (Section 1.7.6). Content is focused on supporting a deterministic review process (Section 1.7.7).

[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 Section C.2 Reference lifecycle model for hazard analysis in Appendix C.

[7] Or reviews that their agents or third parties perform.

1.7.1 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 in this section.

1.7.1.1 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 that are rooted in systemic causes and those that are contributed to through engineering deficiencies in system development). To quote from [6]:

> 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 reevaluated through learning cycles before broader application of RIL-1101.

It is assumed that hazards internal to the DI&C system that are contributed by hardware elements are well understood. Therefore, review of hardware-related HA is addressed in Section 2.7 only briefly[9].

1.7.1.2 Extrapolation from recent licensing experience

Subject matter (e.g., the 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 indicated a trend. It is possible that new issues[10] will surface in upcoming reviews that were not explicitly addressed in RIL-1101. Its adequacy should be tested through several learning cycles before it is applied more broadly.

1.7.1.3 Support for application-specific customization of SRP Chapter 7

Selection and extent of treatment of subject matter in this document was further narrowed to support customization of Chapter 7 of the Standard Review Plan (SRP) [7] specifically for the needs foreseen for the mPower project.

1.7.2 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.

---

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

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

[10] Example: Hazardous scenarios in systems using FPGA or CPLD platforms for implementation.

1.7.3 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.

Within the scope of the SRP Chapter 7, the scope of RIL-1101 is further limited to the digital safety automation (the DI&C equipment), including hazards from interaction with its environment. The operator, the operator-automation interface, and the associated control room are treated as part of the environment (Section 2.4.1) of the system in scope.

1.7.4 Focus on safety-related systems for NPPs

Prevalent public standards [8] and guides ([9], [10], and [11]) 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, an 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 depends (see Appendix K).

Review of analysis for hazards external to the DI&C system, in general, is covered by parts of NRC's standard review plan beyond the part applicable to the DI&C systems [7]. 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 2.4.1 and Sections E.4 and E.5 of Appendix E), because such hazards are not new considerations.

1.7.5 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 depends (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 2.4.1).

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

1.7.6 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 2.1 through 2.6, 2.8, and 2.9).

Systemic causes represent a special kind of common cause of failure[11] (CCF) because their propagation is often 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 caused by the breakdown of a component in a hardware system is relatively easier to identify and analyze.) In

[11] "Failure" in this context means "loss of the top-level safety goal."

a system with complex logic[12], recognizing and understanding the cause-and-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 2.4.2, 2.6.1,and 2.6.3). Some can be recognized through analyzing interactions of the system with its environment (e.g., Section 2.4.1). Many other dependencies occur through organizational processes (e.g., Section 2.2), technical processes (e.g., Sections 2.3, 2.4.3, 2.5, 2.6.2,and 2.6.4), and supporting or auxiliary processes. RIL-1101 does not enumerate all contributory factors and relationships exhaustively, but uses examples of 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.

1.7.7 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] Contributory hazards originating in the system-development lifecycle or rooted in systemic causes[15] are pervasive (permeating) in their effects. The governing variables are not sufficiently controlled in the current state of practice to even 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.

1.7.8 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[16], as it is 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 inspections, tests, analyses, and acceptance criteria (ITAAC) (depicted in the upper right sector of Figure 1) verify that these requirements and constraints have been satisfied. These verification activities are not a part of reviewing hazard analysis as it is 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, firmware, or software) has met the requirements and constraints

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

[13] Scope also excludes qualitative classification or gradation.

[14] Exception: Section 2.7 pertaining to hardware components.

[15] The focus of RIL-1101.

[16] Figure 1 is a simplified depiction; see its note.

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 it is delineated in RIL-1101.

5. The safety analysis report (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 requirements and constraints 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 in the development lifecycle of each of its elements.
2. Iteration at any phase might reveal that the phase has introduced a new hazard.
3. The corrective action might simply be a revision within that phase or it might require a change in a preceding phase, invalidating the result of the preceding phase.
4. The latter case might require multiple iterations and tradeoffs, making the analysis correspondingly more difficult.
5. Verification and validation (V&V) activities during the mainstream system development are also iterative (from discovery of an anomaly through identification of root cause(s) and performance of corrective action on the artifact to performance of corrective action on the process), with each change generating another iteration. Examples of activities included in corrective actions include the following:

   5.1. Identifying an additional constraint,

   5.2. Making an assumption explicit, and

   5.3. Formulating a task to validate an assumption.

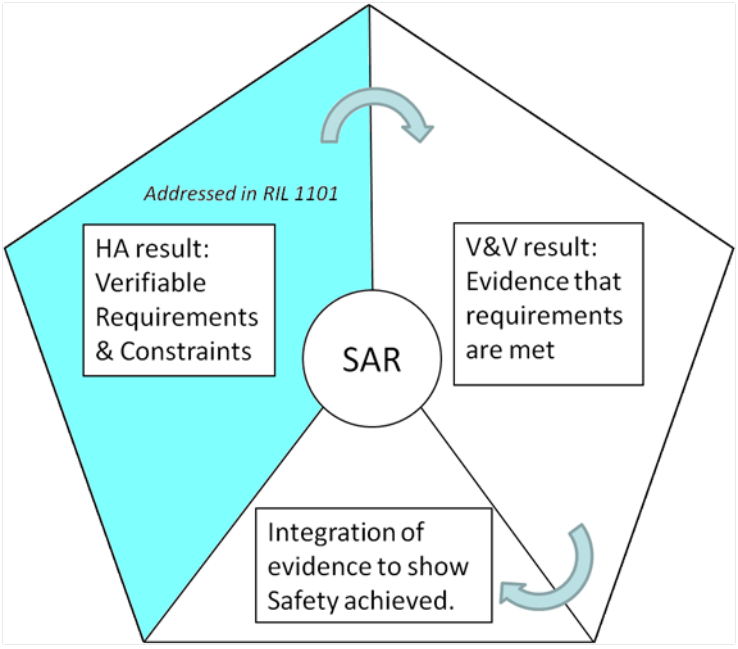

Figure 1: Relationship of HA-evaluation scope in RIL-1101 to overall safety analysis.

1.8 Organization of RIL-1101

Section 2 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 item H-0-1G of Table 1 in Section 2.1, summarizes the state of the art in HA.

Section 2 is organized by groups of contributory hazards, as explained below.

1. These groupings foster different perspectives on (or projections of) intertwined[17] issues, and are not intended to be mutually exclusive partitions.
2. Sections 2.1 through 2.3 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. Sections 2.4 through 2.9 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 title of each table (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, a left-hand cell includes an example of a scenario[18] illustrating some hazard space.
   - 5.3. A right-hand 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 in the headings of the tables' "ID" columns forms the start of the label for the values in each row and is not repeated in any row. Examples are "H-0-", "H-culture-", and "H-OTproc-"; the heading "H-0-" would be combined with the value "7" in a cell below it to derive the full label "H-0-7". The prepended symbol represents the thematic relationship of items within a table, as conveyed in the caption of the table. For example, in Table 4, the symbol "H-OTproc-" prepended to an item ID in the left column of a row indicates that it is a scenario of contributions to hazards through the organization's technical processes. Similarly, in Table 3 "H-culture-" indicates that it is a scenario of contributions to hazards through the organization's culture.

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

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

5.4.2. The “<i>” portion of the label is a numeric character that is 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.

5.5.2. In this manner, a common prepended symbol represents the corresponding commonality in the relationship of a scenario of contributory hazards and conditions which reduce the corresponding hazard space.

6. Hyperlinks enclosed in square brackets [] 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; and
   6.3. between conditions reducing the various hazard spaces.
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. Not all of the many-to-many relationships are hyperlinked.
10. Where needed, a note structure, distinguished by indentation, font type, and font size provides a brief explanation or example for an “H-alpha-<i>” or “H-alpha-<i>-G<j>” paragraph.
11. Enclosure within braces {} in a cell indicates that the symbol prepending the enclosure applies to each enclosed item.
12. Enclosure within parentheses () in a cell indicates a reference for more information on the item in the cell. For example, in Table 1, cell ID H-0-3G1, the enclosure (in common position (CP) 2.1.3.1 in [12]) indicates that the cited reference has more information about the condition, “the requirements … are validated.”
13. A link to an item in an appendix leads to further elaboration and background.

Section 3 explains how HA review fits in the regulatory framework.

Section 4 summarizes the contribution of RIL-1101 and Section 5 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.

2 CONSIDERATIONS IN EVALUATING HAZARD ANALYSIS

RIL-1101 primarily addresses factors contributing to the degradation of a safety function that are rooted in engineering[19]. These factors are part of a network of causes or dependencies (see Appendix K) that result in some deficiency in the system (or deficiency in prevention of interactions), which could lead to the degradation of a safety function. RIL-1101 refers to these factors as contributory hazards.

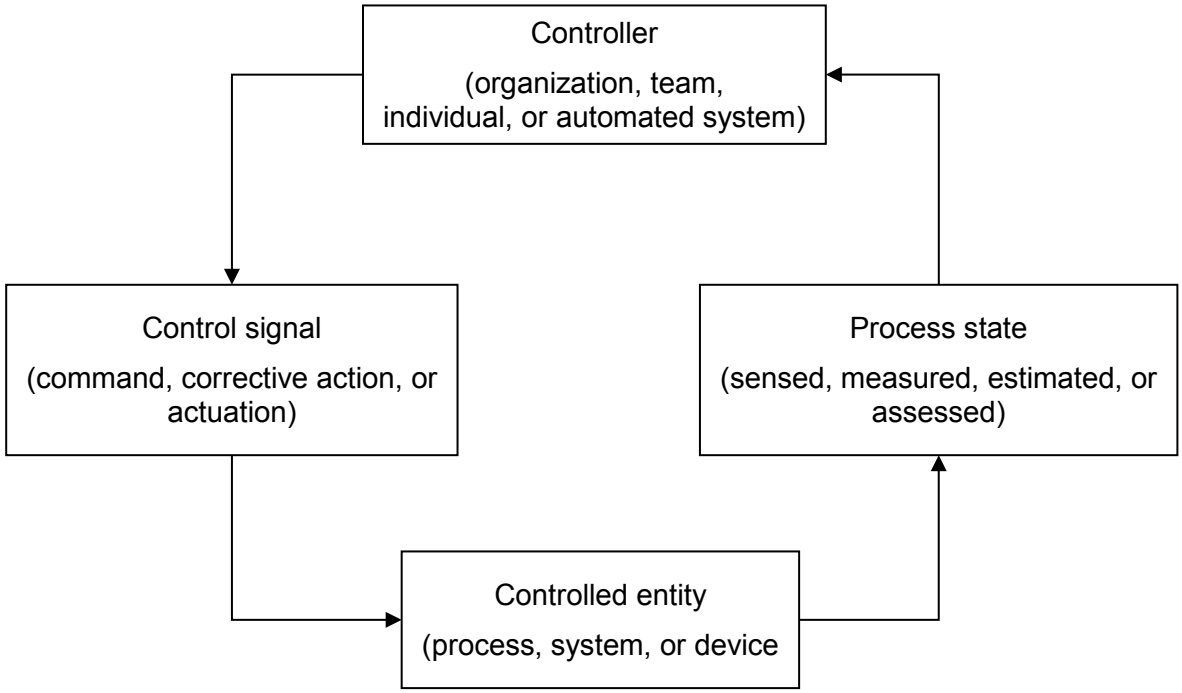

Figure 2: Example of a dependency structure (cyclic graph).

However, recent experience has revealed that propagation paths of hazards are not always linear and that cause-and-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, [13] 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) [14][15] and failure modes and effects analysis for design (DFMEA) [16][17], do not support the discovery of such contributory hazards well. RIL-1101 is intended to address these gaps.

Experience with complex systems in general [18], and with digital systems for critical functions in diverse application sectors in particular, has revealed that common practice does not ensure the absence of conditions contributing to hazards.

The difficulties that the NRO experienced (as it reported to the Advisory Committee on Reactor Safeguards (ACRS) [19]) are examples of the more general trends of increasing system complexity and increasing contribution of systemic causes towards malfunctions. Generally

[19] As opposed to factors arising from random hardware failures during operation.

accepted engineering standards[20] do not provide sufficiently specific guidance to ensure their technically consistent and efficient application to digital systems with such complexity. Such reviews require significant additional information from the applicant, significant additional review effort, and reliance on judgment to address the gap in the existing review guidance. These gaps were identified in [20] as sources of uncertainty 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.

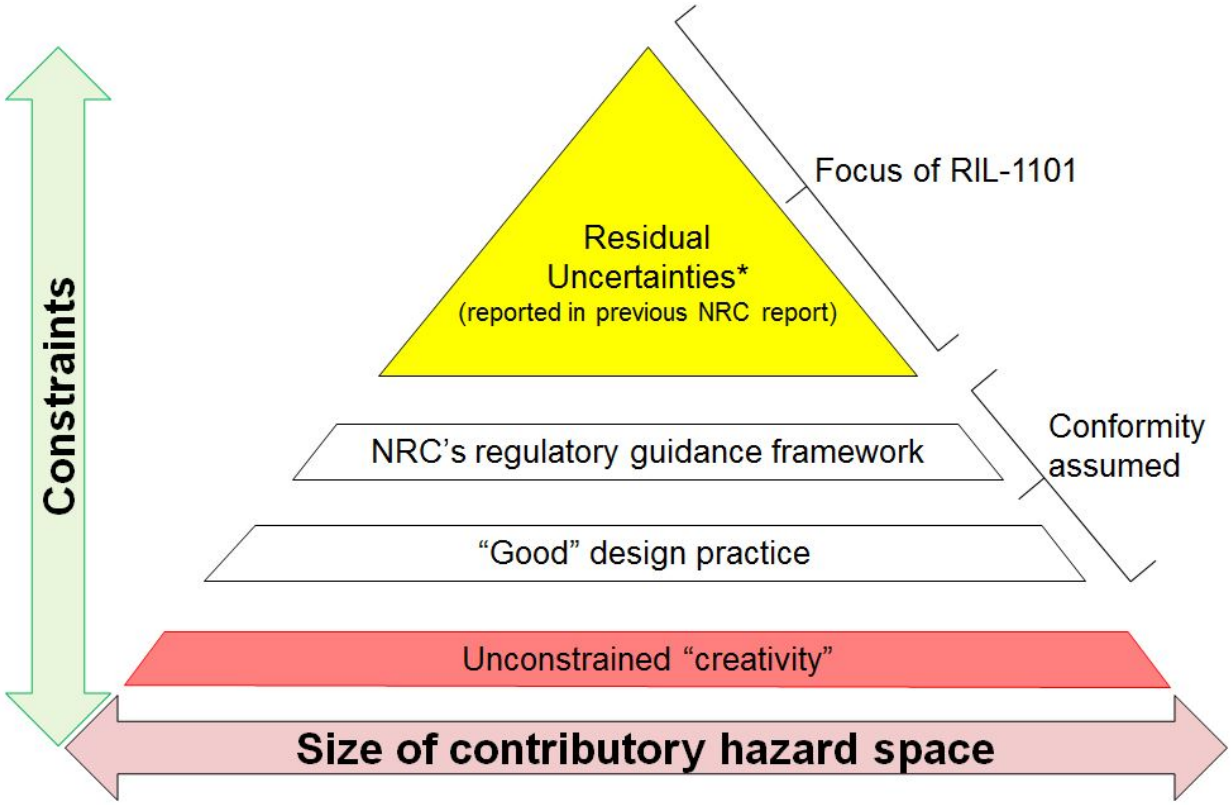

Figure 3: Contributory hazard space in focus.

2.1 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[21] being analyzed. Recognizing that, often, hazardous conditions are obscure and difficult to identify, the performers (typically a team) should have the requisite[22] competence. The controls that correspond to the hazardous conditions should be adequate. The analysis should flow down to all the elements and factors

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

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

[22] Proficiency only in FMEA for random hardware failures might not suffice.

on which the safety function or its integrity depends. Table 1 includes such overarching considerations in evaluating the HA of a digital safety system. Because 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 hazards		**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 or 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 is validated, including elements on which it depends (see Appendix K) [H-0-8↓, H-0-9↓], and including the resulting requirements and constraints, (in CP 2.1.3.1 in [12]) independently, without exacerbating H-culture-9. Also see Appendix F.3. 1. The HA-validation team has the requisite competence [H-0-2G1]. 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).

Contributory hazards		Conditions that reduce the hazard space	
ID H-0-	**Description**	**ID H-0-**	**Description**
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. [H-0-7G1↓]
7	Flowdown from the controls [H-0-6-G1↑] to verifiable requirements and constraints is inadequate.	7G1	Requirements and constraints [H-0-6G1↑] are formulated and validated to be correct, complete, and consistent in consideration of preference[23] order 1 through 4 as follows: 1. Prevent hazard 2. Eliminate hazard 3. Contain hazard (prevent propagation) [H-SR-4G4↓] 4. Monitor, detect and mitigate[24] 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)[25] [H-SR-4G5↓] 4.5. (Recipient[26] 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 on which the safety functions allocated to the system depend. See [H-Dep-1.1↓] in Table 2.	8G1	All dependencies (see Appendix K and Section C.1.3 of Appendix C) 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 [H-Dep-2↓] and [H-Dep-3↓], in Table 2.	9G1	All dependencies are identified and analyzed to confirm 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.

[23] Based on the extent of reduction of hazard space, potential fault space, and uncertainty space.

[24] To maintain a safe state.

[25] For example, the operator or another automated device or system.

[26] For example, the operator or another automated device or system.

Contributory hazards		Conditions that reduce the hazard space	
ID H-0-	**Description**	**ID H-0-**	**Description**
10	Propagated effect of changes introduces inconsistencies, invalidating previously performed HA.	10G1	Starting from the initial HA performed on the functional concept (in CP 2.1.3.2.3 in [12]), the HA is revised at every phase[27] of the development lifecycle, with change-control management and configuration management. Examples of contributory hazards that might 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 [H-0-8G1↑]: 1. No added monitoring, detection, mitigation or other requirement has introduced some new hazard. 2. No complexity-increasing side effect from the change has introduced some other yet-unanalyzed hazard.
10.1	Hazard-introducing effect of iterations is not well understood.	10.1G	H-0-9G1↑ H-0-10G1↑ H-0-10G2↑
11	Required hazard-control action is degraded.	11G1	Each required control action is analyzed for ways in which it can lead to the hazard, such as: 1. A control action is not provided; for example: 1.1. Data sent on a communication bus is not delivered. 2. A control action is provided when it is not needed. 3. An incorrect state transition occurs (e.g., a combination of items 4 and 5 below). 4. An incorrect value is provided; for example: 4.1. Invalid data is provided. 4.2. Stale input value is treated inconsistently. 4.3. An undefined type of data is provided. 4.4. Data is provided in an incorrect message format. 4.5. Incorrect initialization occurs. 5. A control action is provided at the wrong time or out of sequence. 6. A control action is provided for too long a duration (e.g., for continuous-control functions). 7. A control action is provided for too short a duration; for example: 7.1. A signal is deactivated too early (e.g., for continuous-control functions). 8. A control action is intermittent when it is required to be steady; for example: 8.1. Chatter or flutter occurs.

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

Contributory hazards		Conditions that reduce the hazard space	
ID H-0-	Description	ID H-0-	Description
			8.2. A pulse or spike occurs. 8.3. Degradation is erratic. 9. A control action interferes with another action; for example: 9.1. It prevents access to a needed resource; for example: 9.1.1. It is affected by the "babbling idiot" problem. 9.1.2. It locks up and does not release the resource 9.2. It corrupts needed information. 10. A control action exhibits 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 CP 2.1.3.2.7 in [12]) and corresponding requirements & constraints are derived (see checklist in Appendix H for examples).

As HA evaluation progresses further, the selection of information from Sections 2.2-2.9 will be case-specific, depending on the nature of the object and completeness of product-based analysis.

2.1.1 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 and show relationships to later items in this RIL.

H-0-{6, 7, and 11}: These factors address the flowdown from direct hazards to system constraints to required controls to verifiable requirements and constraints. Sections 2.2 through 2.9 elaborate on hazard contributors encountered in the flowdown.

H-0-{8 and 9}: Whereas "ineffective hazard recognition" has been recognized as a serious issue [6], unrecognized dependencies (see Table 2 and Section C.1.3 of Appendix C) become 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.

H-0-8: The extent of dependencies in a system and its elements might not be fully understood or might not be understood in the same way by all parties engaged in developing the system; alternatively, 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, that the dependencies are correctly understood, that the conditions that might degrade a safety function (external and internal) are identified, and that the commensurate constraints are formulated (see Table 2).

2.1.2 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.

H-0-9: The extent of dependencies on processes, including the physical processes in the plant, might 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 on the process activity and factors on which that activity depends. This process dependency structure can be applied to organize and understand the contribution of organizational processes (Section 2.2) as well as technical processes (Section 2.3). This process dependency structure is also applicable to any other creative but deterministic activity from which predictable, verifiable, and analyzable results are needed. Each activity step is affected by the procedures and resources employed in performing that activity. As shown in Figure 4, examples of resources include some tool, other aid, information or competence (e.g., H-culture-6G3) of the performer as symbolized with the cross in a circle. Note that information, tool, or other aid could also depend upon competence. The quality of the work product depends on the quality of the procedures and resources and on their use; that is, any deficiency is a contributory hazard. The following are examples that indicate less-than-adequate controls and thus less-than-adequate understanding of interdependencies across processes.

1. Organizational processes lack such controls; or
2. The organization does not apply such controls to the feeder processes, food chain, or supply chain; or
3. The organization does not plan for such understanding at the system-concept phase of the lifecycle.
4. Adequacy of competence of indirectly employed human resources (in Figure 4, symbolized with the dotted cross in a dotted circle) are not validated.

H-0-10.1: 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[28] could change the system concept or design, requiring another HA cycle to evaluate the impact of such changes. The cumulative and cascading effects of these iterations might not be well understood, with the potential to miss subtle implications of a change.

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

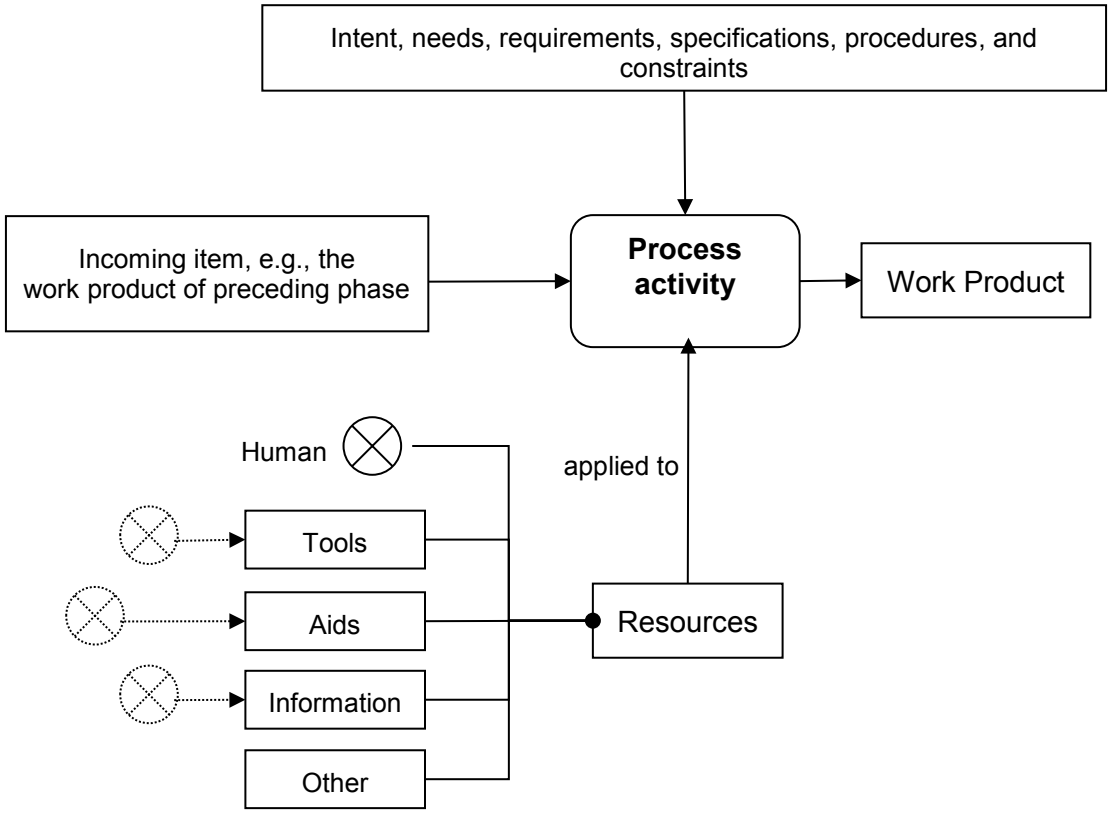

Figure 4: Factors influencing the work product of development.

Table 2: Examples of contributions to hazards through interdependencies

Contributory hazards		Conditions that reduce the hazard space	
ID H-Dep-	**Description**	**ID H-Dep-**	**Description**
1	**Unrecognized interdependencies in the system:** Interdependencies 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. [H-0-8↑]	1G1	All interdependent systems, elements, processes, and factors affecting a safety function are identified. See H-culture-{8G2 and 9G2}.
		1G1.1	Design rationale is recorded and tracked. See Appendix F.2.
		1G2	The items identified in accordance with H-Dep-1G1 are configuration items.
		1G3	The interdependencies or relationships among these items are unambiguously described, especially those affecting emergent behavior. Also see H-culture-{12G2 and 12G3}.
		1G4	Semantics of the relationships are explicit: Relationships might not merely be sequential (chained) or tree structures, but also cycles (often feedback control loops).[29] [H-ProcState-2G1↑]
		1G5	The interrelationships of these configuration items are identified (e.g., by means of an overall NPP-level architecture).

[29] Contrast this situation with the typical chain of events initiated by the failure of a hardware component.

Contributory hazards		Conditions that reduce the hazard space	
ID H-Dep-	Description	ID H-Dep-	Description
		1G6	These interrelationships are also a configuration item or set of configuration items.
		1G7	Independent verification assures that these configuration items represent reality. [H-0-8.1G1↑]
		1G8	The effects of these dependencies are 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 an explicit analysis of the impact of change. (Generalized from CP 2.7.3.1.5 in [12].). See Appendix F.2.
		1G10	The change-impact analysis is independently verified.
		1G11	The change-impact analysis is a configuration item.
1.1	Dependencies through the environment of the digital safety system are not recognized; for example: • Dependencies on the physical processes • Dependencies on degraded behavior of related instrumentation and peripheral equipment	1.1G1	The effects of these dependencies are analyzed to prove that the safety function is not degraded.
		1.1G2	H-culture-8G2
2	**Unrecognized interdependencies in the development process:** Interdependencies in the system-development process, feeder processes, supporting processes, elements, and environments are not understood, leaving some vulnerability, which can lead to a deficiency in the system, which could in turn lead to the degradation of a safety function. [H-0-9↑]	2G1	All interdependent 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 and 9G2}.
		2G2	These are configuration controlled items (henceforth, configuration items).
		2G3	The interdependencies or relationships among these items are unambiguously described, including cycles created through feedback loops[30]. Also see H-culture-{12G2 and 12G3}.
		2G4	The interrelationships 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.
		2G5	Some combination of independent assessment, audit, and verification ensures that these configuration items accurately represent reality.
		2G6	Any change in any of these configuration items is managed through a change-control process.
		2G7	The effects of these dependencies are analyzed to prove that the safety function is not degraded.
		2G8	H-culture-8G2

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

Contributory hazards		Conditions that reduce the hazard space	
ID H-Dep-	Description	ID H-Dep-	Description
3	Dependencies through supporting services and processes are not recognized.	3G1	The effects of these dependencies are analyzed to prove that the safety function is not degraded. See H-culture-{8G2 and 9G2}.
		3G2	H-culture-8G2
4	Dependencies through resource[31] sharing are not recognized; examples might be: • Contention for the shared resource • Corruption of the resource (e.g., data)	4G1	The effects of resource sharing are analyzed to prove that the safety function is not degraded. See H-culture-{8G2 and 9G2}.
Note: Whereas "ineffective hazard recognition" has been recognized as a serious issue [6], unrecognized dependencies are an increasing contributor to this issue because 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. Also see Appendix K.			

2.2 Evaluation of hazard analysis—organizational processes

Organizational processes include management processes, infrastructural processes, and other supporting processes. The term "supporting processes" includes the change-impact analysis process and maintenance processes on 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[32] 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.Table 3 identifies some common concerns.

Table 3: Examples of contributions to hazards through an organization's culture

Contributory hazards		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 [21].)	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.

[31] Examples might be: Skilled resources for development and computing memory or processor time during execution.

[32] This aspect of HA roughly corresponds to, but is significantly broader than, the HA mentioned in Table 1a in [10].

Contributory hazards		Conditions that reduce the hazard space	
ID H-culture-	**Description**	**ID H-culture-**	**Description**
		1G3	The organization has integrity.[33]
		1G3.1	The process[34] 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 ensures accountability for effective achievement of safety.
		2G1.1	Influencing factors are organized in an effective control structure[35] (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 a "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 concerns) is rewarded. 3. See Appendix F.4.2 and F.4.4.
		4G3	Supporting communication and decisionmaking channels exist and the management encourages their usage (e.g., an individual can express safety concern directly to those ultimately responsible). See Appendix F.4.2.
		4G4	Each identified hazard is logged and tracked to its closure, as explained in subsections C.3.2 and C.3.3 of Appendix C. See Appendix F.

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

[34] Applicable to any activity in any process in the organization that is influenced by its management.

[35] A comprehensive safety-governance structure that includes the higher levels of management.

Contributory hazards		Conditions that reduce the hazard space	
ID H-culture-	**Description**	**ID H-culture-**	**Description**
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 [21])	5G1	Safety and quality issues are discovered and resolved at 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 quality and quantity of the required resources are not planned or allocated in a timely manner.	6G1	Resources required[36] are estimated with adequate accuracy[37] in a timely manner.
		6G2	The required resources are allocated in time.
		6G3	Skilled resources have the necessary competence to perform the assigned activity. [H-0-2G1; H-SRE-1G{1, 2, and 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 for mistakes, thereby increasing the potential fault space or contributory hazard space. (Adapted from Annex B in [21].)	7G1	Run critical cognitive tasks to completion (and make this the default practice of the organization). Interruption is allowed only when the task has progressed to a stable, well-understood state, so that the interruption does not increase the hazard space.
8	Processes do not produce deterministic, predictable results.	8G1	A defined, documented, and 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

[36] Such as the type of competence, degree or level of competence or proficiency, and the amount of effort and time.

[37] Implied constraint: Processes are adequately designed and controlled. [H-0-9G1, H-culture-8G1, and H-OTproc-1G]

Contributory hazards		Conditions that reduce the hazard space	
ID H-culture-	**Description**	**ID H-culture-**	**Description**
		8G2	The organization follows disciplined communication and cognitive processes to achieve collective mindfulness, to know when to adjust and adapt the standardized processes, and to learn from the shortcomings. See Sections F.2 and F.4 of Appendix F.
9	When system lifecycle activities are distributed across multiple organizations or parts of the same organization, safety-relevant information[38] is not communicated efficiently, letting key items of information "fall through the cracks." See note at end of table. [H-SRE-7↓]	9G1	Cross-organizational dependencies are understood clearly. Also see H-culture-8G2.
		9G1.1	The organization maintains cross-organizational connections that improve collective mindfulness (for example, by 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 and 12G3}.
		9G3	Decomposition of safety goals from NPP level analysis and allocation to safety-related systems is complete, correct, consistent, and unambiguous.
10	Mistakes are repeated.	10G1	Continuous improvement is integral to all processes. See Section F.4 in Appendix F.
11	Heavy dependence on testing[39] 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[40] and designed to prevent and discover safety and quality issues as early in the development lifecycle as possible. See H-culture-{12G2 and 12G3} and Section F.2 of Appendix F.
12	Dependence on implicit information (including implicit assumptions). [H-ProcState-4↑] and [H-SR-11↓]	12G1	All information on which assurability of safety depends is explicit and configuration-controlled.

[38] Implied constraint: H-0-9G1

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

[40] Examples of work products: The safety plan, quality plan, and V&V plan, including plan for demonstrating completeness of coverage.

Contributory hazards		Conditions that reduce the hazard space	
ID H-culture-	Description	ID H-culture-	Description
		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 situational awareness of the context and involves the cultivation of diverse perspectives. See Appendix F.
		12G3	The organization establishes a system for tracking the bases and premises of engineering decisions. See Section F.2 of Appendix F and Appendix J.
Note for H-culture-9: The quality of cross-disciplinary, cross-organizational communications 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.			

2.3 Evaluation of hazard analysis—technical processes

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

- Requirements engineering—see Section 2.5.
- Architecture engineering—see Section 2.6.
- Detailed design—see Section 2.8.
- Implementation—see Section 2.9.
- 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 4 (adapted from Appendix A.1 in [20]), premised on the satisfaction of constraints identified in Table 3.

Table 4: Examples of contributions to hazards through technical processes

Contributory hazards		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 such a level of detail 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 the results of such work elements correctly [H-culture-8G1↑]. Also see [H-culture-8G2].

Contributory hazards		Conditions that reduce the hazard space	
ID H-OTproc-	**Description**	**ID H-OTproc-**	**Description**
2	Any process variable in any work element might contribute to some deficiency, if it is not adequately controlled. [H-OTproc-1↑]	2G	Each process variable in each work element is controlled and supported with the proper methods, tools, and competence to execute that work element correctly and to integrate the results of such work elements correctly. [H-OTproc-1G↑] (Figure 2 and 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[41] of personnel available to perform that activity. Also see [H-culture-6G3].
3.1	Difficulty of understanding the architecture (if it is inadequately described, for example) is a contributor to the cognitive load.	3.1G1	The system architecture is analyzable and comprehensible. [H-OTProc-3G1↑], [H-S-1.1G1↓], and [H-S-2G6↓].
4	Mistakes occur[42] (leading to deficiencies in the system); however, technical processes are not designed with the necessary robustness and resilience to protect them from such mistakes.	4G1	The organization's technical processes include processes to detect and recover from mistakes (e.g., verification and audit).
		4G2	[H-culture-8G2].
5	The organization believes incorrectly that its processes are adequate, exposing it to unknown sources of deficiencies for which it cannot identify the causes.	5G1	The process is assessed and certified independently.
		5G2	Qualified independent resources assess the process. [H-culture-6G1 and H-culture-6G2]
		5G3	[H-culture-8G2].
6	The processes in real-life execution deviate from the designed processes, resulting in exposure to unknown sources of deficiencies, for which it cannot identify the causes.	6G1	[H-culture-{1G3.1 and 2G1.1}
		6G2	The process in execution is audited independently.
		6G3	Qualified resources are available to audit the process.
		6G4	[H-culture-6G4 and H-culture-8G2]. Also see and Appendix F.4.
7	Less accumulated experience and reusable results than there are for the systems of the previous generation; 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	[H-0-9G1] H-culture-{2G1.1 and 8G1} H-OTproc-{1G and 2G}
		7G2	More rigorous analysis—see Table 1 and Table 3. Appropriately conservatively derived requirements and constraints.
		7G3	[H-culture-{8G2, 12G2, and 12G3}]

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

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

Contributory hazards		Conditions that reduce the hazard space	
ID H-OTproc-	Description	ID H-OTproc-	Description
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 [18] of systems, processes, and organizations (involving people from multiple organizations, multiple disciplines, and 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 2.5; HA results in safety requirements & constraints). • Architecture engineering (elaborated in Section 2.6). • Software engineering (elaborated in Sections 2.6.4 and 2.8).			

2.4 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 the concept is based. Sometimes, the associated requirements are embodied in a "concept of operations" document. Sometimes HA[43] of a functional concept is called preliminary hazard analysis (PHA[44]); 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 2.1) is most effective in the concept phase of a system-development lifecycle. Avoidance of these contributors to hazards (see Table 1 and Tasks T1 through T3 in Table 21) requires clear description and tracking of the evolving system concept and its relationship with its environment, as discussed in this section.

2.4.1 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 5, Table 6, and Table 7. These tables also identify conditions that reduce the respective hazard spaces.

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

Section 2.4.1.1 includes examples of hazards related to interactions with the plant processes.

Section 2.4.1.2 includes examples of hazards related to interactions with instruments, controls, and networks in the system's environment.

[43] It roughly corresponds to but is significantly broader than the HA mentioned in Table 1b in [10].

[44] The concept and PHA are good candidates for discussion with the applicant before it submits the license application.

Section 2.4.1.3 includes examples of hazards contributed through the human-interaction aspect of the system's environment.

Section 2.4.2 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.

2.4.1.1 Hazards related to interaction with plant processes

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 state of 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 5 shows examples of contributory hazards and conditions that reduce the respective hazard space.

Table 5: Examples of contributions to hazards through interactions with the plant

Contributory hazards		**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[45] 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[46] 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 so 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, 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.

[45] Examples: Pressure, temperature, flow, and neutron-flux density.

[46] Examples: Energy conversion, equipment degradation, and component degradation.

Contributory hazards		Conditions that reduce the hazard space	
ID H-Proc State-	**Description**	**ID H-Proc State-**	**Description**
		1.2G3	Signal indicating event of interest is not missed because of inadequate sampling, as determined through mathematical analysis.
		1.2G4	Capturing event of interest does not disrupt any other action on 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, and networks) to represent reality with fidelity[47].
3	Allocation of safety functions and properties from a system at a higher level of integration to a system at a lower level is not correct, complete, or consistent, or is ambiguous.	3G1	Relationships with losses of concern are identified, and commensurate safety goals are explicitly formulated, in NPP-level analysis.
		3G2	Decomposition of safety goals into required safety functions (design bases) is complete, correct, consistent, and unambiguous.
		3G3	Allocation of safety requirements to safety-related systems[48]) 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 Table 8 in Section 2.5.1.1.
		3G5	The boundary of the system being analyzed is well-defined with respect to its environment (in CP 2.1.3.2.1 in [12]).
		3G6	The interface to and interactions with the plant are specified and constrained in such a manner that the system is understandable [H-S-2↑], verifiable[49] [H-S-1.1], and free from interference [H-SA-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, and functions 4.2. Procedures specifying 4.1

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

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

[49] That is, satisfaction of the constraint or specification is verifiable by analyzing the system concept.

Contributory hazards		Conditions that reduce the hazard space	
ID H-Proc State-	**Description**	**ID H-Proc State-**	**Description**
		3G7	Constraints on other elements in the environment of the system are explicit.
		3G8	Restrictions & constraints placed on the system are explicit; example constraints might be: 1. Compatibility with existing systems. 2. Compatibility with physical and natural environment. 3. 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 item 3 of Appendix A.3 to [20].) [H-culture-12↓]	4G1	See: H-ProcState-3G7, H-culture-{12G2 and 12G3}, and Appendix J.
		4G1.1	[H-culture-12G1↓] The organizational processes (Section 2.2) 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 and 12G3}.
		4G1.2	If an assumption is found to be invalid or there is a change from the previous assumption: 1. A corresponding change-impact analysis is performed. 2. The affected part of the HA is repeated. 3. Commensurate changes in constraints or requirements are identified. 4. An analysis of the impact of those changes is performed. 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 to 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) might 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 might not be well-defined, leading to appropriate considerations "falling through the cracks."			

Figure 5 depicts a "progressive"[50] migration from a 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 must be monitored. Intervention must be completed within this (yellow) region.

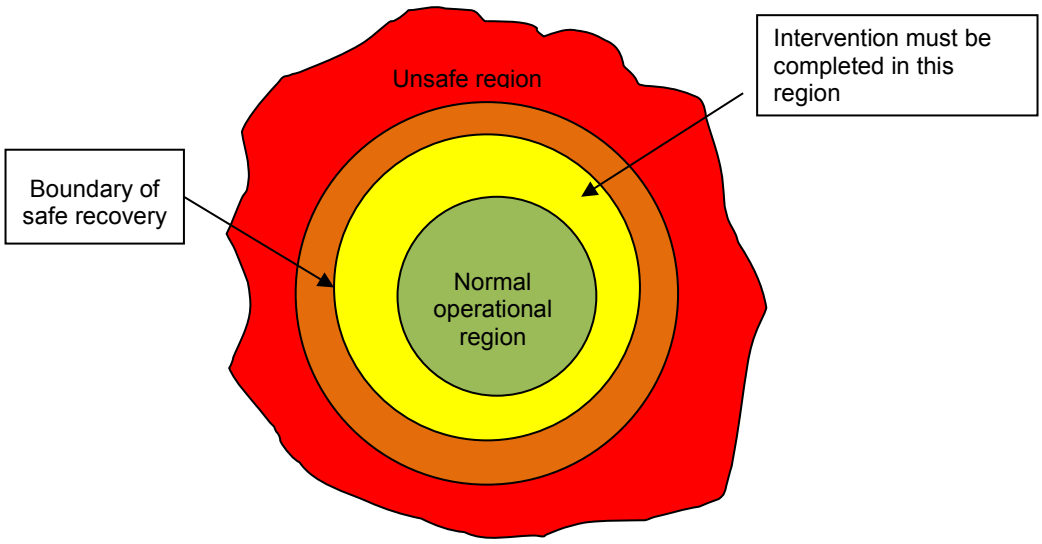

Figure 5: Regions of state space for hazard analysis.

2.4.1.2 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.

[50] Under the premise that degradation is not sudden or unpredictable and that its progression can be monitored.

Constraints on the NPP-wide I&C architecture are derived from the quality[51] attributes or properties of the safety-related system being analyzed. Quality attributes are discussed in Section 2.5.1.1, including in Table 8, which also applies to the NPP-wide I&C architecture.

> Note: Criteria for the evaluation of HA for 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 as exemplified in Table 3 through 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.

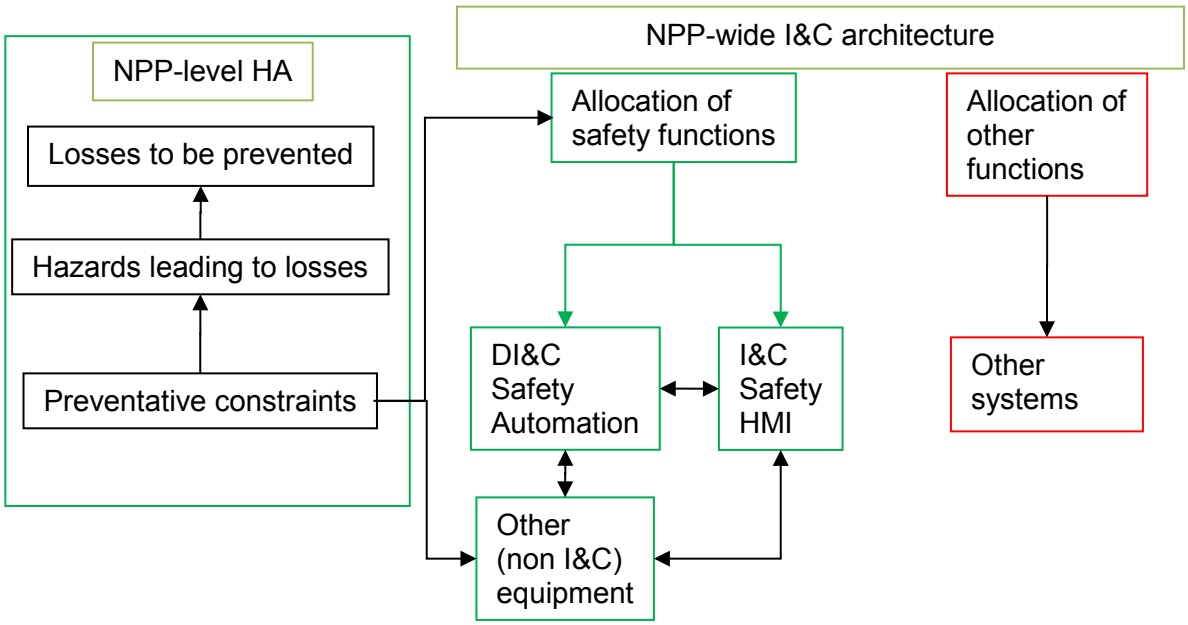

Figure 6: NPP-wide I&C architecture—allocation of functions in the concept phase.

2.4.1.3 Contributory hazards from human/machine interactions

Hazards of the kind grouped in Table 1 through Table 5 could also affect human/automation interactions.

The tables in this section supplement those with some examples of more specific hazards contributed through human/automation interactions (Table 6) and through inadequacies in the associated engineering (Table 7).

[51] Other terms for these properties are “quality-of-service (QoS) properties” and “non-functional requirements.”

Table 6: Examples of contributions to hazards through human/machine interactions

Contributory hazards		Conditions that reduce the hazard space	
ID H-hmi-	**Description**	**ID H-hmi-**	**Description**
1	Inconsistency between human-perceived process state and real process state.	1G1	Process state presented to the human accurately represents the real physical state in value and time.
2	Inconsistency between human-perceived state of an instrument[52] and real state of the instrument.	2G1	Instrument (e.g., actuator) state presented to the human accurately 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 (that is, the 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	Either inconsistent behavior of automation is avoided or automation detects its inconsistency and notifies human.
		3G5	Unintended[53] 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.
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 [6] [9] [18].	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 human/automation interaction) is described clearly, including behavior and human/automation interaction under all combinations of off-normal conditions (e.g., in the presence of a fault).

[52] For example, a sensor or actuator.

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

Table 7: Examples of contributions to hazards through human/machine interaction engineering

Contributory hazards		Conditions that reduce the hazard space	
ID H-hmiP-	Description	ID H-hmiP-	Description
1	Loss of information across disciplines (e.g., automation engineering, human-factors engineering, and control-room design). [H-culture-9↑ and H-SR-3↑]	1G1	System is engineered holistically, including crosscutting analysis. (Adapted from footnote 82 in Appendix A.3 to [20].)
2	Confusing human/machine interface design.	2G2	H-hmi-3G3.
3	Cognitive overload.	3G3	H-OTproc-3G1.

2.4.2 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 Section 2.4.1.2).

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 interrelationships 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 [item 2↑ of H-SA-3G3 in Table 13].
2. Freedom from interference between a monitoring element and its monitored element [item 4↑ of H-SA-3G3 in Table 13].
3. Compromise of redundancy through a dependency (e.g., input data or resource sharing). Also see items H-0-8 and H-0-9 in Table 1.
4. Compromise of redundancy in the concept of voting[54] logic.

The conditions (to reduce the respective hazard spaces) provided in Table 8 and Table 13 apply recursively to the most finely grained level of the system architecture and recursively to the most finely grained level of the software architecture. These conditions also apply to the mappings (e.g., through composition and decomposition) from one level to another in the architecture hierarchy[55] and through all stages of derivation of requirements & constraints and the subsequent development lifecycle stages (e.g., detailed design and implementation).

2.4.3 Contributory hazards from conceptualization processes

Some hazards contributed through weaknesses in the cultural and general technical processes of the organization (Table 3 and Table 4), which were introduced in Section 2.2, strongly apply to the concept phase of the system-development lifecycle.

Requirements engineering (Section 2.5) and architectural engineering (Section 2.6) apply to the concept phase also—see Table 12, Table 13, and Table 14.

[54] Example: In a quad-redundant system for a space system, four computers were connected by a multiplexer/demultiplexer module. A diode in the interconnections failed in such an unanticipated way that the condition was not sensed in the same way by the four computers. (In footnote 84 in Appendix A.3 to [20].)

[55] The mapping could contribute a hazard because some abstractions can mask problems.

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

2.5 Evaluation of hazard analysis—Requirements

Experiences of many critical application domains have shown that identifying valid requirements for a critical digital safety system is one of the weakest links in the overall engineering process. 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 [20]. 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.

2.5.1 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 2.1 (overall hazard analysis and understanding dependencies leading to loss events) and 2.4.1 (understanding hazards in relation to the environment of the safety system, including hazards contributed from inadequate definition of the boundary of the safety system, from invalid assumptions (see Appendix J), and from interactions with 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 2.5.1.1 introduces the concept of associated quality requirements. Section 2.5.1.1 also introduces the concept of derived quality characteristics or requirements in an organizing framework, known as a "quality model" [22]. Section 2.5.1.2 identifies some common weaknesses in formulating verifiable requirements and Section 2.5.1.3 identifies some common weaknesses in the associated requirements-engineering processes.

2.5.1.1 Quality requirements

Figure 7 shows two categories of requirements – functional and quality requirements [22]. In general, a functional requirement may be associated with one or more quality requirement or constraint; for example, in the context of this RIL, SAFETY and SECURITY are top-level quality requirements, which depend upon other constraints or required characteristics, as shown in Figure 8 (the connectors represent dependency relationships).

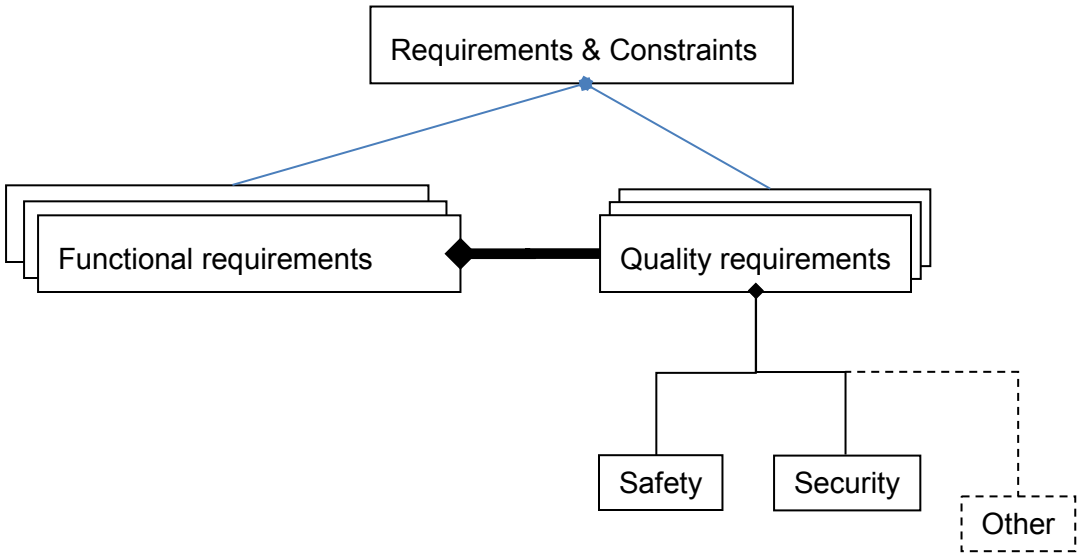

Figure 7: Quality requirements should be explicit.

In a regulated environment, the SAFETY property should be assurable independently. Then, as shown in Figure 8, "Assurability" is a required attribute[56]. Figure 8 also shows other quality attributes upon which "Assurability" depends. 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), which eliminate from further consideration those system concepts that do not satisfy these constraints. In other words, these constraints reduce the hazard space in the design space. Table 8 shows the logical derivation of these constraints, corresponding to the relationships shown in Figure 8; the reasoning is explained below:

1. To be able to assure that a system is safe, one must be able to verify that it meets all of its safety requirements. [H-S-1]
2. For a system to be verifiable, it must not be possible for one element of the system to interfere with another. [H-SA-3]
3. If the conceived system is too complex, adequate verification is infeasible. [H-S-1.1]
4. If it is too complex, one cannot understand it. [H-OTproc-3G1]
5. If one cannot even understand it, how can one assure that it is safe? [H-S-2]
6. Verifiability is a required system property, flowing down from the system to its elements (constituents) and progressing to the most finely grained element; it implies corresponding verifiable specifications. Verification also includes analysis at various phases in the development lifecycle, well before[57] an artifact is available for physical testing. Examples of conditions for verifiability include:
   6.1. Ability to create a test (or verification) case to verify the requirement.
      6.1.1. Observability

[56] It distinguishes regulated safety systems from non-critical systems such as those for entertainment.

[57] This is known as "static analysis" when it is performed on a computer program (code). 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]

6.2. Ability to constrain the environment of the object of verification.

7. For "analyzability," the system must have predictable and deterministic[58] behavior. [H-S-1.2]

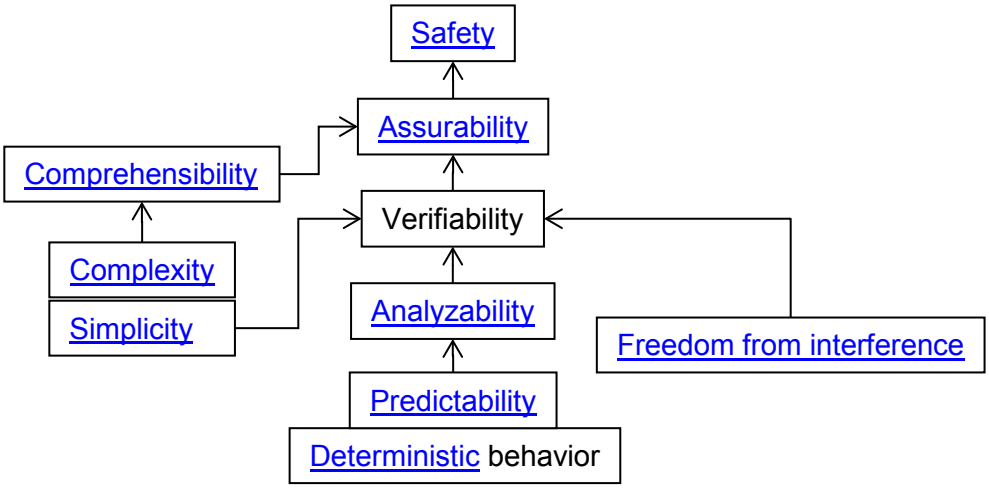

Figure 8: Quality characteristics to support safety.

Table 8: Examples of contributions to hazards through quality attributes

Contributory hazards		Conditions that reduce the hazard space	
ID H-S-	**Description**	**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 project's limitations).	1G1	Verifiability is a required system property, flowing down from the system to its constituents and progressing to the most finely grained element. (Adapted from CP 2.2.3.11 in [12].) [H-S-1.1G1↓]
		1G1.1	Verifiability of a work product is checked at every phase of the development lifecycle, at every level of 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[59] 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 item 4 in Section A.4 of Appendix A to [20]).
		1.1G1.2	The flowdown ensures that: 1. Allocated behaviors satisfy the behavior specified at the next higher level of integration. 2. Unspecified behavior does not occur.

[58] Yields deterministic results.

[59] Defined as complexity that is not essential to support a safety function.

Contributory hazards		Conditions that reduce the hazard space	
ID H-S-	**Description**	**ID H-S-**	**Description**
		1.1G1.3	The behavior of the system is composed of the behaviors of its elements in such a way that when all of the elements are verified individually, their compositions may also be considered verified[60]. This property is satisfied at each level of integration, flowing down to the most finely grained element in the system.
		1.1G1.4	Development follows a refinement process.
1.1.1	Unanalyzed or unanalyzable conditions exist. For example, not all system states, including unwanted ones such as fault states, are known and 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 that lead to failure modes are known (in the first item of CP 2.2.3.14 in [12]). 3. The safe-state space of the system is known (in the second item of CP 2.2.3.14 in [12]).
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[61]. [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[62] monitoring and indication): 1. Initiation of function 2. Completion of function (in the last item of CP 2.1.3.4 in [12]) 3. An intermediate state, where one is needed to maintain a safe state in case of a 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 combinations of inputs and transition conditions.
		1.3G2	A hazardous condition can be detected in time to keep the system in a safe state. (in the third item of CP 2.2.3.14 in [12]).

[60] No unspecified behavior emerges.

[61] Does not yield deterministic results.

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

Contributory hazards		Conditions that reduce the hazard space	
ID H-S-	Description	ID H-S-	Description
2	Comprehensibility: System behavior is not interpreted correctly and consistently 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. (See note) Also see H-culture-{12G2 and 12G3}.
		2G2	The system is simple enough (not too complex) to be understood in the same meaning by its community of users.
		2G3	Behavior is understood or interpreted completely, correctly, consistently, and unambiguously by different users interacting with the system. Also see H-culture-{12G2 and 12G3}.
		2G4	The allocation of requirements to some function and the allocation of that function to some element of the system are bidirectionally[63] traceable. (in item 2 of Section A.4 of Appendix A to [20]).
		2G5	The behavior specification avoids mode confusion, especially when functionality is nested (in item 3 of Section A.4 of Appendix A to [20]).
		2G6	The architecture is specified in a manner (e.g., through its language and structure) that is unambiguously interpretable by the community of its users (e.g., reviewers, architects, designers, implementers), that is, the people and the tools they use (in item 9 of Section A.4 of Appendix A to [20]).
Note for H-S-2G1: Sometimes, experts can understand implicit meaning. However, explicit information is needed for machine interpretation (i.e., through tools).			

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 and decomposition, including all phases of the software development lifecycle. Commensurate architectural constraints are identified in Section 2.6.

2.5.1.2 Contributory hazards through inadequate system requirements

Activities leading to identification of functional requirements for safety were introduced in Sections 2.1 (overall hazard analysis, including understanding of dependencies leading to a loss event or degradation of a safety function) and 2.4.1 (understanding hazards in relation to the environment of the safety system, including hazards contributed from inadequate definition of the boundary of the safety system, from invalid assumptions (see Appendix J), and from interactions with other systems and people). Table 9 identifies further contributory hazards resulting from weaknesses in identifying and formulating requirements. The content of Table 9 is

[63] This does not imply that one-to-one relationships are necessary.

adapted mostly from Section A.3 of Appendix A to [20]; other sources are cited within the respective item in Table 9. For hazards contributed through weaknesses in interfaces and interactions across elements of the system, see Section 2.6.1.

Table 9: Examples of contributions to hazards through inadequate system requirements

Contributory hazards		**Conditions that reduce the hazard space**	
ID H-SR-	**Description**	**ID H-SR-**	**Description**
1	Mistakes occur because the environment is misunderstood.	1G1	[H-SRE-{1G1, 1G2, and 1G3}↓]. See H-culture-{4G1, 4G2, 4G3 and 6G3}; Subsections C.3.2 and C.3.3 of Appendix C; and Appendix F, especially Section F.1.
2	Input constraints are misunderstood or improperly captured. [H-SR-1↑]	2G1	[H-SRE-{1G1, 1G2, and 1G3}↓]. See H-culture-{4G1, 4G2, 4G3 and 6G3}; Subsections C.3.2 and C.3.3 of Appendix C; and Appendix F, especially Section F.1.
		2G2	Criteria for input validation are correctly established. See Sections F.2 and F.4 of Appendix F.
3	Incomplete requirements.	3G1	See Table 1.
		3G2	H-ProcState-3G5.
		3G3	HA includes interactions with the environment of the system; see Section 2.4.1.
		3G4	Interrelationships and interactions with the environment are analyzed in all configurations and modes (including degraded ones) and through all changes from one mode to another. [H-SR-3G3↑]
		3G5	In HA at the system concept phase (Section 2.4), an architectural model or representation of the system (functional or behavioral) concept includes a (functional or behavioral) model or representation of the environment, especially of the physical processes (Appendix H in [23]). [H-SR-3G3↑ and H-SAE-{1G1, 2G1, 3G1, and 4G1}↓]
		3G6	Process behavior models[64] (H-SR-3G5) include identification of safe-state regions and the trajectory[65] of safely recoverable process states. See Figure 5 and [23].
		3G7	Process behavior models (H-SR-3G5) include time dependencies, relationships and constraints. [14] . Also see Table 5 and Appendix I.
		3G8	[H-SRE-{1G1, 1G2, and 1G3}↓].

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

[65] The state space within which recovery is provable.

Contributory hazards			Conditions that reduce the hazard space
ID H-SR-	**Description**	**ID H-SR-**	**Description**
4	Inadequate protection or defense against latent faults. [H-SR-3↑]. Note tension with [H-SR-20].	4G1	Monitoring: Feasible trajectories[66] of appropriate state variables[67] or parameters and expected values are known and monitored. (Generalized from CPs 2.1.3.2.3 and 2.1.3.2.4 in [12].)
		4G2	Detection: Appropriate parameters of the system or element are monitored to detect departure from safe state (e.g., by applying discriminating[68] logic on the monitored parameters) in conjunction with predictive behavior models, but considering [H-SR-{19 and 20}]
		4G3	Intervention: On detection of departure from safe state, intervention is performed to keep the plant in safe state. (Adapted from CP 2.2.3.7 in [12].)
		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 overload[69].
		4G6	[H-SRE-{1G1, 1G2, and 1G3}↓]
5	Inadequate identification of sources of uncertainty, their effects, and their mitigation. [H-SR-3↑]	5G1	[H-SRE-{1G1, 1G2, and 1G3}↓]
6	Deficiency in requirements for fault containment. [H-SR-3↑]	6G1	[H-SRE-{1G1, 1G2, and 1G3}↓]
7	Inadequate or improper generalization to capture classes of issues.	7G1	[H-SRE-{1G1, 1G2, and 1G3}↓] See H-culture-{4G1, 4G2, 4G3, 4G4, and 6G3} and Appendix F, especially Section F.1.
8	Inconsistent requirements.	8G1	[H-SRE-{1G1, 1G2, and 1G3}↓]
9	Inadequate protection or defense against invalid input. [H-SR-4↑]	9G1	See H-SR-2G2 The validity of the value of each input is monitored (in CP 2.1.3.2.4 in [12]).
		9G2	Intervention on detecting invalid input is specified in order to keep the system in a safe state.
10	Instrumentation errors are uncorrected or inadequately compensated for.	10G1	Required calibrations and corrections are known and applied (generalized from CP 2.1.3.2.5 in [12]). [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 item 3 in Appendix A.3 to [20]). See Appendix J. [H-culture-{12G1-12G3}↑]
12	Invalid assumption about the environment.	12G1	See Appendix J, H-culture-12G3, Appendix C.3.3, and Appendix F.2.

[66] For example, values over time and rates of change.

[67] Include inputs and outputs.

[68] For example, detect infeasible or unexpected values.

[69] A single event may cause many deviations crossing monitoring thersholds. However, if their communication is not managed, it may overload system resources or the operator.

Contributory hazards		Conditions that reduce the hazard space	
ID H-SR-	**Description**	**ID H-SR-**	**Description**
		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	The record of each assumption [H-SR-12G1] includes the consequences if the assumption turns out to be false. (In item 4 in Appendix A.3 to [20].) Also see Appendix J.
		13G2	Requirements include measures to mitigate the consequences of assumptions that fail to hold. (In item 4 in Appendix A.3 to [20].)
		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 on the associated methods and tools). (In item 3.1 in Appendix A.3 to [20]) See: Appendix J and H-culture-{12G2 and 12G3}. Examples: 1. Requirements from the application software on system platform services (hardware (HW) and software (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	The record of each assumption [H-SR-12G1] includes how and when it will be validated. (In item 3 in Appendix A.3 to [20].)
15	Incorrect order of execution or timing behavior. [H-ProcState-1.1] [H-ProcState-1.2]	15G1	An explicit, verifiable (as determined through mathematical analysis) specification of the order of execution and timing interrelationships; the specification includes considerations for multiple concurrent physical processes, inter-process synchronization, and shared resources (in CPs 2.1.3.2.2 and 2.2.3.5 in [12]). See Appendix I.
16	Interrelationships and interdependence across requirements are not clearly understood or recognized [H-0-6 through H-0-8], 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.

Contributory hazards			Conditions that reduce the hazard space
ID H-SR-	**Description**	**ID H-SR-**	**Description**
17	Interference from unintended interactions or side effects. [H-S-1↑]. See Table 2.	17G1	Interactions are limited provably[70] to those required for the safety functions.
		17G2	Verifiable constraints are specified to prevent unspecified behavior or side effects.
18	Effects of sudden hardware[71] failure, especially failure of 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↓]
19	Allocated set of requirements leads to conditions that are unanalyzable or difficult to analyze.	19G1	[H-SRE-{1G1, 1G2, and 1G3}↓]
20	Adding backups (or fault protection) can introduce new hidden dependencies and impair analyzability. [H-SR-19↑]	20G1	[H-SRE-{1G1, 1G2, and 1G3}↓]
21	Layers of protection against software faults might impair analyzability. [H-SR-19↑]	21G1	[H-SRE-{1G1, 1G2, and 1G3}↓]
22	Inability to correctly integrate elements of a system (e.g., subsystems, hardware components, or software components). [H-SR-{1, 2, 3, 8, 12, 13, 15, 16, and 19}↓] [H-SwR-2↓] [H-SRE-7↓] [H-SwRE-1↓] [H-HwP-1↓]	22G1	[H-SRE-{1G1, 1G2, and 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 that its correctness can be concluded 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[72] 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 the safe state. See Figure 5. Also see Appendix F.
Note for H-SR-16G1: Relationships may be one-to-one, one-to-many, many-to-one, and many-to-many.			

[70] It is possible to show that unspecified interactions cannot occur.

[71] Also see Table 17.

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

2.5.1.3 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: Examples of contributions to hazards through inadequate system-requirements engineering

Contributory hazards		Conditions that reduce the hazard space	
ID H-SRE-	**Description**	**ID H-SRE-**	**Description**
1	Inadequate competence. [H-culture-6↑]	1G1	The team engaged in these activities is a group with 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 of the engineering lifecycles. (See note). 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 through 6}]. See H-culture-6G3 and item 5 of Appendix F.1.
2	Ambiguity in the natural-language textual description. [H-SAE-2↓]	2G1	A subset of the natural language is used in order to unambiguously describe requirements to the community of users[73] of the system and the requirements. This subset features, for example: 1. A closed set of language elements. 2. Unambiguous semantics for each language element. 3. Unambiguous compositions of language elements and compositions of compositions. Also see H-culture-{12G2 and 12G3}. [H-SAE-1G1↓ and H-SAE-1G2↓]
		2G1.1	Formal properties are abstracted for later use in verification of the next phase of the work product. [24][25]

[73] Users include people employed in creation, modification, interpretation, transformation. maintenance, V&V, and regulation (adapted from the last sentence of CP 2.3.3.1.1 in [12]) and the tools they use (i.e., the language should be unambigious to the tools for the functions allocated to them).

Contributory hazards		Conditions that reduce the hazard space	
ID H-SRE-	Description	ID H-SRE-	Description
		2G2	The language subset (H-SRE-2G1) supports distinct identification and description of the following: 1. Assumptions about the environment [23]; Appendix J. 2. Input from the environment such as a command (that is, some signal requiring a state-changing effect plus required behavior), query, process state, or other data. 3. Output (for example, some signal having state-changing effect, or state notification, or 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 ([14] and Appendix I) and quality-of-service (QoS) constraints. 10. Criteria to monitor and detect violation of a constraint [21].
3	Incorrect formalization from intent or natural-language text	3G1	[H-SRE-2G1↓ and H-SRE-2G2↓] [H-SAE-1G1↓ and H-SAE-1G2↓]
		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 CP 2.1.3.2.4 in [12]). Also see Table 1 and H-culture-{12G2 and 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 CP 2.1.3.1 in [12] and in item 2 in Section A.4 of Appendix A to [20]). [H-S-2G4↓]
		6G2	Each requirement is a configuration-controlled item.[74]

[74] Other relevant references: IEEE 828, "IEEE Standard for Configuration Management in Systems and Software Engineering," and IEEE 1042, "IEEE Guide to Software Configuration Management."

Contributory hazards		Conditions that reduce the hazard space	
ID H-SRE-	Description	ID H-SRE-	Description
7	Loss of information[75] across disciplines, processes, and organizational units (e.g., system engineering, software engineering, hardware engineering, safety, and quality). [H-culture-9↑] [H-SR-3↑] [H-SwRE-1↓]	7G1	Systems are engineered holistically, including crosscutting analysis. (Adapted from footnote 82 in Section A.3 of Appendix A to [20].) See H-culture-9G1, H-culture-9G2, and Section F.1 of Appendix F.
		7G1.1	The interaction across a system or an element and its environment is identified explicitly. Example: Models at every level of integration are compatible with one another and information can be integrated and analyzed across the various models. See H-culture-{12G1, 12G2, and 12G3}.
Note for H-SRE-1G1: The types of expertise that the team will require might differ from phase to phase.			

2.5.2 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, and consistent unambiguous requirements for software are supposed to flow down from the system engineering lifecycle, typically (in practice) V&V for these properties occurs from the software engineering perspective[76] 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 2.6 for constraints on software architecture. Contributory hazards and constraints identified in Section 2.6.1 for the system architecture also apply to software. Derived constraints on software design and implementation are included in Sections 2.8 and 2.9.

[75] Current practice divides systems engineers, software engineers, and hardware engineers; often failures occur due to gaps between these specialties. (From footnote 82 in Section A.3 of Appendix A to [20].)

[76] For example, the software engineer checks the correctness of understanding and elicits additional information in order to make the requirements explicit and unambiguous.

2.5.2.1 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: Examples of contributions to hazards through inadequate software requirements

Contributory hazards		Conditions that reduce the hazard space	
ID H-SwR-	Description	ID H-SwR-	Description
1	Inadequate flowdown of properties (Table 8) and other constraints from the system engineering lifecycle (Table 9). [H-SwR-2↓]	1G1	Corresponding constraints (Table 8 and Table 9) are derived and applied to software.
2	Inadequate flowdown of requirements and constraints to support integration of elements into a correctly working system.	2G1	Corresponding constraints (Table 8 and Table 9) are derived and applied to software.
3	Inadequate flowdown of requirements and constraints from NPP level to the safety system and then to its elements, including software.	3G1	Where requirements and constraints are decomposed or derived from upstream (source) requirements, their composition satisfies the source requirements and does 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[77] for anomalous behavior of software (adapted from CP 2.3.3.1.5 in [12]), but do not introduce the adverse conditions identified in H-SR-{19 and 20}.

2.5.2.2 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: Examples of contributions to hazards through inadequate software-requirements engineering

Contributory hazards		Conditions that reduce the hazard space	
ID H-SwRE-	Description	ID H-SwRE-	Description
1	Loss of information across disciplines, processes, and organizational units (e.g., system engineering, software engineering, hardware engineering, safety, and quality) caused by the division of organizations, people, and work along disciplinary lines. [H-culture-9↑]	1G	H-SRE-{7G1 and 7G1.1}↑. Also see H-culture-{9G1, 9G2, 12G2, and 12G3} and Appendix F.

[77] It is a diversely redundant defense against deficiency in requirements to prevent the anomaly.

Contributory hazards		Conditions that reduce the hazard space	
ID H-SwRE-	Description	ID H-SwRE-	Description
2	Loss of information across disciplines caused by the use of 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 and 3G1}↑] [H-HwP-5G1↓] See Appendix F.4.

2.6 Evaluation of hazard analysis—Architecture

System failures traceable to architecture rank high in incidence in 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 [20].

2.6.1 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 interrelationships (i.e., interfaces, interconnections, and interactions), whether these are direct or indirect, intended or unintended, explicit or implicit, static or dynamic, or "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 as exemplified in Table 3 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: Examples of contributions to hazards through interference

Contributory hazards		Conditions that reduce the hazard space	
ID H-SA-	Description	ID H-SA-	Description
3	A system, device, or other element (external or internal to a safety system) might affect a safety	3G1	[H-SR-17G1↑]

Contributory hazards		Conditions that reduce the hazard space	
ID H-SA-	Description	ID H-SA-	Description
	function adversely through unintended interactions caused by some combination of deficiencies, disorders, malfunctions, or oversights. [H-SR-17↑]	3G2	Interactions and interconnections that preclude complete[78] V&V are avoided, eliminated, or prevented. (CP 2.2.3.11 in [12])
		3G3	Freedom from interference is assured provably[79] across: 1. Lines of defense [26]. 2. Redundant divisions of system (CP 2.2.3.6 in [12]). 3. Degrees of safety qualification[80] (CP 2.2.3.3 in [12]). 4. Monitoring & monitored elements of the system.
		3G4	Analysis of the system demonstrates that unintended behavior is not possible.[81] 1. Interaction across different sources of uncertainty is avoided. 2. The architecture precludes unwanted interactions and unwanted or unknown hidden couplings or dependencies (in item 6 in Section A.4 of Appendix A to [20]). 3. Specified information exchanges or communications occur in safe ways (in item 6 in Section A.4 of Appendix A to [20]).
		3G5	Only well-behaved interactions are allowed [H-S-1.2G{1, 2, and 3} and H-S-1.3G{1 and 2}↑]
		3G6	Constraints are identified for such contributing hazards from the environment as electromagnetic interference; see examples in Section E.4. of Appendix E.
		3G7	The impact of dependency-affecting change is analyzed to demonstrate no adverse effect.
4	[H-SA-3G4↑]: A function, whose execution is required at a particular time, cannot be performed as required because of interference through sharing of some resource it needs.	4G1	Analysis of the execution-behavior of the system proves that such interference will not occur. For example, worst-case 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 on which their satisfaction depends. Example: A sampling interval that characterizes the monitored variable with fidelity.

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

[79] By, for example, showing that there is no pathway by which such interference could occur.

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

[81] By, for example, showing that there is no pathway by which such unintended behavior could occur.

Contributory hazards		Conditions that reduce the hazard space	
ID H-SA-	**Description**	**ID H-SA-**	**Description**
		5G1.1	The proper required sampling interval is determined through mathematical analysis.
		5G1.2	Discretization and digitization do not affect the required fidelity determined through mathematical analysis.
		5G1.3	Aliasing is avoided.
		5G1.4	Sampling periods to monitor discrete events are determined correctly through mathematical analysis.
6	Sampling and update intervals are not appropriate for the timing constraints of the associated control actions. [H-SR-15]	6G1	Update intervals support the timing constraints of the required control actions determined through mathematical analysis.

2.6.2 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 2.7, 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: Examples of contributions to hazards through inadequate system architectural engineering

Contributory hazards		Conditions that reduce the hazard space	
ID H-SAE-	**Description**	**ID H-SAE-**	**Description**
1	The architecture[82] description (including requirements allocated to its elements) is ambiguous, rendering it vulnerable to interpretations other than those intended. For example, textual descriptions use words and expressions (and graphic representations use symbols) for which commonly understood meanings have not been agreed on by the community using this information. [H-S-2G6↑] [H-SAE-2↓ and H-SAE-3↓]	1G1	The description method supports distinct, unambiguous description of the following: 1. Assumptions about the environment. 2. Input from the environment such as a command (some signal requiring state-changing effect plus required behavior), query, or data. 3. Output (that is, some signal having state-changing effect) or 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. Interrelationships of elements. 7. Interactions required across elements. 8. Constraints on the behavior and interactions of each element; e.g., timing constraints (Appendix I) and QoS constraints. 9. Criteria to monitor and detect violation of a constraint.

[82] 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, and function/procedure architecture.

Contributory hazards		Conditions that reduce the hazard space	
ID H-SAE-	**Description**	**ID H-SAE-**	**Description**
		1G2	The language (graphic or text-based) used in the description or specification is unambiguous; for example, it has: 1. A closed set of language elements. 2. Unambiguous semantics for each language element. 3. Unambiguous semantics for the compositions (e.g., rules of composition) of language elements and their compositions.
		1G3	The method and language are applied correctly.
2	Transformation 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 (a) system conceptual or requirements level to (b) system design level to (c) software design level to (d) software implementation level to (e) 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 results in unintended behavior caused by 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 results in unintended behavior caused by 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	5G1	Each tool is qualified for safety-grade use.

Contributory hazards		Conditions that reduce the hazard space	
ID H-SAE-	Description	ID H-SAE-	Description
	engineering is not qualified to produce, manipulate, or handle a safety-grade architectural artifact (e.g., system, element, and data).	5G2	Restrictions necessary for safe use of a tool are identified and the set of restrictions are 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 might 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, which is tracked as a configuration-controlled item.
		6G2	Restrictions on individual tools, their information-exchange functions, and their interactions (all of which are needed for safe use of the tools as a set) are identified and the set of restrictions is 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 (perhaps because semantics are not preserved in the flowdown of specifications or their realization).	7G1	Each pre-existing element is qualified for the environment[83] 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.
		7G2	Restrictions on the use of a pre-existing element in the target environment are identified and the set of restrictions are 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, and H-SR-13G3
8	Individuals performing architectural engineering functions might 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, and know the usage limitations of the tools, elements, and artifacts available to them; the limitations, for example, might be incomplete V&V, known defects and deficiencies, limited conditions of use, and unvalidated assumptions.

[83] Including assumptions about the environment—also see H-culture-12.

2.6.3 Contributory hazards in software architecture

The information in Section 2.6.1, Table 8, and Table 13 also applies[84] to software architecture, especially to 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 interrelationships (i.e., interfaces, interconnections, and interactions), whether these are direct or indirect, intended or unintended, explicit or implicit, static or dynamic, "normal" or "abnormal."[85]

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[86], and to transformations from one development phase or level to another.

Table 15: Examples of contributions to hazards through software architecture

Contributory hazards		Conditions that reduce the hazard space	
ID H-SwA-	Description	ID H-SwA-	Description
1	Software contributes to or exacerbates the 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[87] element is a composition of the behaviors of its constituent elements, with well-defined and unambiguous rules of composition.[88] (In item 5 in Section A.4 of Appendix A to [20].) 1. Interfaces of elements include specification of their behavior, and are unambiguously specified (adapted from CP 2.3.3.2.2 in [12]). 2. Interactions across elements occur only through their specified interfaces; that is, the interactions adhere to principles of encapsulation (adapted from CP 2.3.3.2.2 in [12]).
		1G2	The system is modularized using principles of information hiding and separation of concerns, avoiding unnecessary interdependence (in item 7 in Section A.4 of Appendix A to [20]).
		1G2.1	Corresponding specifications are modularized.
		1G2.2	Corresponding specifications, plans, and procedures for verification are modularized.

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

[85] Examples might be invalid input, a hardware malfunction, or a human mistake.

[86] Examples might be the subsystem, module, and subroutine levels.

[87] Non-atomic means that the architecture identifies its subdivisions - it is not the finest-grained element defined in the architecture.

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

Contributory hazards		Conditions that reduce the hazard space	
ID H-SwA-	Description	ID H-SwA-	Description
		1G3	Each element (e.g., a software unit) is internally well-architected, satisfying conditions identified in Table 13, H-SwA-1G1, and H-SwA-1G2, in such a way that its quality requirements (Section 2.5.1.1) 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 and unambiguous rules of composition. [H-SwA-1G1↑] 2. Paths from inputs to outputs avoid unnecessary coupling. [H-SwA-1G2↑] 3. Unnecessary remembering of state information across execution cycles is avoided. (Adapted from CP 2.3.3.2.8 in [12].)
2	Order of execution or timing behavior is not analyzable correctly because of system complexity	2G1	Complexity-increasing behaviors are avoided [H-S-1.1.1G1↑] while simplicity-increasing features are preferred; for example: 1. Static configuration of tasks[89] to be executed in the operating software (adapted from the second and third bullets of CP 2.4.3.8.1 in [12]). 2. Tasks in execution are run to completion[90] (adapted from the first bullet of CP 2.4.3.8.1 in [12]). 3. Static allocation of resources[91] [H-SA-4G1↑] (generalized from bullets 2 through 4 of CP 2.4.3.8.1 in [12]).
3	Behavior is not analyzable mathematically or analysis is not mechanizable for lack of a semantically adequate paradigm or model underlying the behavior specification or description. [H-SAE-{1, 2, and 3}]	3G1	Behavior specification or description method is based on a semantically adequate, unambiguous paradigm [H-SAE-1G1↑ and H-SAE-1G2↑] supporting association of timing constraints [H-SR-13G4↑], other properties (Table 8↑), hierarchical nesting, and abstraction [H-S-1.1G1↑]. An example paradigm is that of an (extended) finite-state machine (adapted from [27] and from CP 2.3.4.1.1 in [12]).

[89] In this context, a task is a schedulable unit of work. Dynamic creation and destruction of tasks is avoided.

[90] For example, interruption and preemption are avoided or mathematical analysis (See Appendix I) proves the satisfaction of constraints on timing and order of execution.

[91] Resources such as memory (information storage) and processor (execution) time.

2.6.4 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: Examples of contributions to hazards through inadequate software architectural engineering

Contributory hazards		Conditions that reduce the hazard space	
ID H-SwAE-	**Description**	**ID H-SwAE-**	**Description**
1	Loss of information across disciplines (e.g., system engineering, software engineering, and hardware engineering) caused by the division of organizations, people, and work along disciplinary lines [H-culture-9↑].	1G	Software is engineered with the requisite complement of competence – see H-SRE-1G1 H-SRE-7G1↑
2	Loss of information across disciplines caused by the use of incompatible paradigms, methods, and tools across disciplines.	2G	H-SAE-{2G1 and 3G1}↑

2.7 Evaluation of Hardware-Related Hazard Analysis

As in the preceding sections, hardware-related HA is treated in two parts—the product (Table 17: Examples of contribution to hazards) and the process (Table 18: Examples of contributions to hazards through inadequate hardware engineering).

Table 17: Examples of contribution to hazards through hardware

Contributory hazards		Conditions that reduce the hazard space	
ID H-Hw-	**Description**	**ID H-Hw-**	**Description**
1	Failure of hardware leads to unanalyzed conditions [H-S-1.1.1↑] (e.g., an unknown state).	1G1	Only hardware with predictable, well-known, and well-understood degradation behavior is used.
		1G2	Degradation is detectable before failure that could lead to unanalyzed conditions (e.g., an unknown state) [H-S-1.2G3↑]. (Adapted from the first clause of CP 2.2.3.7 in [12].)
		1G3	Safety requirements are specified to keep the system in a safe, known state at all times, in all modes of usage, including maintenance, and including degraded conditions. Safety functions may be online or offline; for example: 1. Monitor hardware conditions [H-SR-4G1↑] through: 1.1. Cyclic or periodic online monitoring or 1.2. Offline surveillance. 2. Detect hardware faults [H-SR-4G2↑]—see H-Hw-1G4. 3. Notify (other automation or a human) of the detection. [H-SR-4G5↑] 4. Intervene (to keep the system in a safe state). [H-SR-4G3↑] 5. Perform preventive maintenance such as scheduled replacements. 6. Provide redundancy, including diverse redundancy. (Items 1 through 4 are adapted from CP 2.2.3.7 in [12])

Contributory hazards		Conditions that reduce the hazard space	
ID H-Hw-	Description	ID H-Hw-	Description
		1G4	Requirements are identified for independent, timely detection of a contributory hazard in an instrument or other element on which a safety function depends. For example: 1. In the case of a bi-stable device, the device can be feasibly be only in one stable state or the other, so 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 of the variables, electric current, displacement, and 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 (that is, their set of values over time) is identified, so that indication of an instrument anomaly can be derived from sensed values in the infeasible region.
2	An anomaly in the state of the process is not recognized, identified, or correctly understood because of inadequacy in instrumentation. [H-SR-23↑]	2G1	Progressive degradation, drift, and such other changes in the behavior of instrumentation are properly accounted for. Ways in which this is done might include: 1. Monitoring and tracking of such phenomena. 2. Compensation. 3. Calibration and recalibration. 4. Allowances (margins) for changes that are not accounted for, are not compensated for, or are unknown. 5. Detection of unacceptable deviation. 6. Appropriate intervention—see items 2 and 4 in H-Hw-1G3.
3	An anomaly in the state of the instrumentation for the safety functions (or in the state of another element in the environment on which a safety function depends) is not correctly understood or recognized.	3G1	The instrument's or element's behavior (including its behavior in fault states) satisfies requisite properties such as those identified in Table 8.
4	Loss or interruption of power.	4G1	See H-Hw-1G3. Continuity of power is monitored.
5	Disturbance in power supply.	5G1	See H-Hw-1G3. Quality of power is monitored.

Contributory hazards		Conditions that reduce the hazard space	
ID H-Hw-	Description	ID H-Hw-	Description
6	Inadvertent alteration of invariant information (e.g., program code or fixed data).	6G1	Invariant information is stored in read-only memory (ROM). (Adapted from CP 2.7.3.3.2 in [12]).
7	Change in hardware that is nominally "equivalent" to replaced hardware (e.g., is supposed to be functionally, electrically, or mechanically interchangeable) causes some subtle change that degrades a safety function.	7G1	Criteria for equivalence are correct and complete. Examples of sources of differences might be: 1. Differences in timing behavior. 2. Differences in signal-noise discrimination. Also see Table 1. 3. Differences caused by replacing fixed logic with programmable logic.
		7G2	The criteria mentioned in H-Hw-7G1 are satisfied.

Table 18: Examples of contributions to hazards through inadequate hardware engineering

Contributory hazards		Conditions that reduce the hazard space	
ID HwP-	Description	ID H-HwP-	Description
1	Loss of information across disciplines (e.g., system engineering, software engineering, and hardware engineering) caused by division of organizations, people, and work along disciplinary lines [H-culture-9↑].	1G1	H-SRE-1G1↑ See H-culture-{4G1, 4G2, 4G3, and 4G4} and Appendix F.
2	Loss of information across disciplines caused by incompatible paradigms, methods, and tools across disciplines.	2G1	H-SAE-{2G1, 3G1}↑
3	Preventative maintenance activities on which a safety function depends are not performed correctly or in time [28] [H-Hw-1G3].	3G1	Maintenance schedules specify the preventative actions explicitly and correctly [H-Hw-1G3↑].
		3G2	These maintenance schedules are treated as safety-related activities (including those for performance, verification, and audit) [Table 1].
4	Preventative protection against age-related degradation is not provided in maintenance plans (generalization from [29]).	4G	[See H-Hw-{1G1 and 1G2}.]
5	Computation is incorrect because of a hardware-software incompatibility	5G1	All interdependent elements are specified and configured correctly.

Contributory hazards		Conditions that reduce the hazard space	
ID HwP-	Description	ID H-HwP-	Description
	caused by change in any of the following: 1. Hardware (e.g., floating-point processor). 2. Operating system (OS) 3. OS library software. 4. Compiler 5. Compiler library software. 6. Algorithm (e.g., formula and data types). 7. Application library software. [H-SwRE-2↑]	5G2	The hardware, software, and transformation are qualified and configured correctly for conformance to the specification mentioned in H-HwP-5G1. (Generalized from CP 2.4.3.5.8 in [12]).
6	Selection of output destination (e.g., actuator) or input source (e.g., sensor) is incorrect (for example, because of 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 first sentence of CP 2.3.3.1.7 in [12]).

2.8 Evaluation of Hazard Analysis related to Software Detailed Design

Review of HA under 10 CFR Part 52, “Licenses, Certifications, and Approvals for Nuclear Power Plants,” is limited to review of work products from the pre-certification phases of the lifecycle (e.g., the plan, concept, requirements, and 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. If that’s the case, 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: Examples of contributions to hazards through inadequate software detailed design

Contributory hazards		Conditions that reduce the hazard space	
ID H-SwD-	Description	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 and 3G1}↑
2	Software contributes to or exacerbates the complexity of the system, making it difficult to verify [H-S-1.1↑] and understand [H-S-2↑]. [H-SwA-1↑]	G2	See H-S-1.1G1 H-S-1.1.1G1 H-S-1.1.2G1 H-S-2G1 H-S-2G3
3	Functions, data items, inputs, outputs, and variables in software are named in such ways that the names become difficult to trace back to system requirements (and further back to	G3.1	Naming conventions and data dictionaries are established for ease of comprehension and bidirectional traceability.

Contributory hazards		Conditions that reduce the hazard space	
ID H-SwD-	Description	ID H-SwD-	Description
	the application domain). (Adapted from CP 2.3.4.1.2 in [12]).	G3.2	Naming conventions and data dictionaries are used consistently.

2.9 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) [30] and Common Weakness Enumeration (CWE) [31] 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: Examples of contributions to hazards through software implementation

Contributory hazards		Conditions that reduce the hazard space	
ID H-SwI-	Description	ID H-SwI-	Description
1	Behavior is not analyzable mathematically or analysis is not mechanizable because of 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, by not using interrupts to cover for inadequately understanding timing behavior of the physical phenomena (Table 1 and H-SR-3G7) or the design and implementation (H-SR-13G4 and 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 do not depend on either the time taken to execute the program or the time (referring to an independent “clock”) at which execution of the program is initiated. (Adapted from [32].)
		2G2	Execution speed does not affect correct order of execution.

3 DISCUSSION OF REGULATORY SIGNIFICANCE

Hazard analysis of a digital safety system[92] could address clause 4.8 in IEEE Standard 603-1991 [1], which is incorporated by reference in 10 CFR 50.55a(h)(3). In clause 4.8 in IEEE Standard 603-1991 (quoted below), 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.

[92] 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.

> 4: 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 in accordance with 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 [3] and corresponding clause 10 CFR 52.47(a)(2) of [4], "… analysis of the structures, systems, and components (SSCs) of the facility, with emphasis on performance requirements, the bases, with technical justification therefor, on 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 could be part of the "analysis…of the major structures, systems, and components…". Hazard analysis of a digital safety system identifies design characteristics, 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 [1], which depends on 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.

It is recognized from recent licensing-review experiences that generally accepted engineering standards[93] are not sufficiently specific to ensure consistent application, given the trends in design characteristics, and unusual or novel design features. These novelties require significant judgment that depends on a 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.

Hazard analysis of a digital safety system could lead to development of principal design criteria, in addition[94] to or overlapping the general design criteria in Appendix A to 10 CFR 50, which provide only minimum requirements.

Hazard analysis of a digital safety system could lead from principal design criteria to design bases, including constraints on the architecture and on design and implementation, in such a way that the performance of the functions and the non-exceedance of the constraints are verifiable later in the system-development lifecycle. These derived requirements and constraints lead to the level of design information to which the following requirement in 10 CFR 52.47, "Contents of Applications; Technical Information," 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

[93] The term "generally accepted engineering standards" is mentioned in 10 CFR 50.34(a)(ii)(B), and includes standards cited in the NRC's regulatory guides.

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

> 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…"

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 flowdown (or dependency) paths from a safety function, because it follows a cause-and-effect course of inquiry and reasoning, originating from potential for degradation of the safety function. This cause-and-effect course of inquiry and reasoning could also support developing specific information required in accordance with 10 CFR 50.34(a)(5) through (8) and 10 CFR 52.47(a)(7) and (19), in those cases for which such information is 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 degradation of a safety function allocated to the system under evaluation. As this approach includes dependency analysis, it also supports identification of contributors to a hazard; for example, the potential for adverse effect on diversity or defense in depth.

4 CONCLUSIONS

This RIL provides the U.S. 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, combined license, 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. These issues arise from engineering deficiencies during the development of a digital safety system. These deficiencies are characterized as contributory hazards. Examples of engineering deficiencies include:

1. Unintended or unwanted interactions are not prohibited.
2. The boundary of the digital safety system being analyzed is not defined adequately.
3. Constraints to control hazards from the top level of a digital safety system are decomposed and allocation incorrectly in the flow down the integration hierarchy; and
4. Corresponding requirements and constraints on technical processes, supporting processes, and organizational processes are not derived correctly and completely.

Although the targeted scope was limited, the result supports a broader purpose. Hazard analysis organizes information along cause-and-effect dependency paths (Table 1, items H-0-8 and H-0-9, and Appendix K) from a safety function to a contributing item and provides a framework for reasoning about the (perceived) deficits (Section C.3.3 of Appendix C). In this manner, it contributes to risk-informed evaluation of the system.

The cause-and-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-cited standards which do not contribute to the safety goal. The applicant could also use this framework to justify alternative ways of satisfying the NRC's regulation in cases in which 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 a 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[95] 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 [33].

[95] 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.

5 FUTURE RESEARCH, DEVELOPMENT, AND TRANSITION

The development of this RIL, including knowledge acquired through reviews by experts, has revealed 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.

5.1 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 the NRC's licensing reviewers. There is a need to make this knowledge more easily deployable in practice, including illustrative examples. Consistent with recommendations in [34] about domain-specific software engineering, the knowledge can be made more accessible through techniques to represent the knowledge of the domain in a form that is easy to find and reuse correctly.

In support of a recommendation by the ACRS I&C Subcommittee, the NRC intends to coordinate its plans with the Electric Power Research Institute (EPRI) in order to share the knowledge base that is common across various stakeholders' activities: System development by the applicant or its supplier, hazard analysis by the applicant, and its evaluation by the regulatory reviewer.

In support of a recommendation by the ACRS I&C Subcommittee, transition plans will include learning cycles (e.g., through pilot applications).

5.2 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 trend in 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, such as tool-supported model-based hazard analysis.

5.3 Harmonization and disambiguation of vocabulary

Differences in vocabulary hamper the NRC's ability to learn from NPP experience elsewhere in the world and from other application sectors. The same terms have different meanings. The same concepts have different terms. Different concepts are combined in different ways, introducing more terms for the combinations. These combinations are not directly or easily comparable. These conditions lead to ambiguities and unnecessarily encumber the tasks of hazard analysis and evaluation.

Technology is available to bridge these communication gaps (e.g., modeling knowledge of the domain, as mentioned in Section 5.1). The NRC will coordinate its plans with EPRI.

5.4 International harmonization

Different regions of the world pursue the same or similar safety goals under different regulatory and guidance frameworks, citing 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 5.3, opportunities exist in exploring international harmonization of the technical basis for evaluating hazard analysis.

5.5 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. Consistent with executive guidance, resources can be leveraged by coordinating future R&D with other federal agencies [35], including capability to systematize hazard analysis at the conceptual phase of the system-development lifecycle [36].

5.6 Analysis earlier in the system-development lifecycle

In the case of new reactors, applications for design certification have included safety analysis of software that is based more on process conformance than rigorous V&V of the software itself. 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 introducing that knowledge in the NPP application domain, building on the R&D mentioned in Sections 5.1 and 5.3 and enabling hazard analysis on an architectural model of the system.

5.7 Risk-informed evaluation

Opportuities exist in applying hazard analysis (for example, modeling and analysis of dependencies on systemic causes) to risk-informed evaluation of systems in which safety-significant conditions can arise from unintended interactions, engineering deficiencies, or other systemic causes.

5.8 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's configuration.

5.9 Integrated organizing 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 1.7.8 (see Figure 1) and Appendix C.3.3 (see 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 Organisation of Economic Co-operation and Development's (OECD's)/Nuclear Energy Agency's (NEA's) Halden Reactor Project, the 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, building on the hazard analysis framework introduced in this RIL. 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 to evaluate modifications[96] after a reactor becomes operational.

5.10 Ideas received through review comments

Suggestions and remaining issues identified in review comments are treated as inputs to the 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 contributed through implementations of field-programmable gate arrays (FPGAs) and complex programmable logic devices (CPLDs).

[96] It could support evaluation for 10 CFR 50.59, "Changes, Tests, and Experiments."

6 ABBREVIATIONS AND ACRONYMS

ACRS	Advisory Committee on Reactor Safeguards
ADAMS	Agencywide Documents Access and Management System
CFR	*Code of Federal Regulations*
CP	common position[97]
CPLD	complex programmable logic device
DI&C	digital instrumentation and control
FPGA	field-programmable gate array
FMEA	fault modes effects and analysis
FTA	fault-tree analysis
EQ	environmental qualification
HA	hazard analysis
HAZOP(S)	hazard and operability studies
HQEO	high-quality engineering organization
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
NRO	Office of New Reactors
PHA	preliminary hazard analysis
QoS	quality of service
R&D	research and development
RAI	request for additional information
RES	Office of Nuclear Regulatory Research
RIL	research information letter
SAR	safety analysis report
SRP	Standard Review Plan
V&V	verification and validation

[97] A term used in [12] for a requirement on which the Task Force for Safety Critical Software has total consensus. This task force consists of regulatory experts from the United Kingdom (UK), Germany, Sweden, Belgium, Finland, Spain, Canada, and Korea.

7 REFERENCES

[1] Institute of Electrical and Electronics Engineers (IEEE), IEEE Standard 603-1991, "IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations," Piscataway, NJ, 1991.

[2] *U.S. Code of Federal Regulations*, "Conditions of construction permits, early site permits, combined licenses, and manufacturing licenses" Section 50.55, Chapter I, Title 10, "Energy" (10 CFR 50.55), available at http://www.nrc.gov/reading-rm/doc-collections/cfr/part050/part050-0055.html.

[3] *U.S. Code of Federal Regulations*, "Contents of Applications; Technical Information," Section 50.34, Chapter I, Title 10, "Energy" (10 CFR 50.34), available at http://www.nrc.gov/reading-rm/doc-collections/cfr/part050/part050-0034.html.

[4] *U.S. Code of Federal Regulations,* "Contents of Applications; Technical Information," Section 52.47, Chapter I, Title 10, "Energy" (10 CFR 52.47), available at http://www.nrc.gov/reading-rm/doc-collections/cfr/part052/part052-0047.html.

[5] U.S. Nuclear Regulatory Commission (NRC), "Instrumentation and Controls – Hazard Analysis," Appendix A (Section 7.0) to the proposed "Design-Specific Review Standard for mPower iPWR Design," May 3, 2013, Agencywide Documents Access and Management System (ADAMS) Accession No. ML12318A200.

[6] Corcoran, W.R., "Hazard recognition for quality, safety, and performance improvement," *The Firebird Forum*, Volume 15, Number 3, March 2012.

[7] NRC, "Standard Review Plan for the Review of Safety Analysis Reports for Nuclear Power Plants: LWR Edition – Instrumentation and Controls," NUREG-0800, Chapter 7 ADAMS Accession No. ML070550074.

[8] U.S. Department of Defense (DoD), "Department of Defense Standard Practice: System Safety," MIL-STD-882E, Washington, DC, May 11, 2012, available at http://www.system-safety.org/Documents/MIL-STD-882E.pdf.

[9] Ericson II, C.A., Hazard Analysis Primer, self-published through Seattle, WA: CreateSpace, February 14, 2012.

[10] U.S. Air Force, Air Force System Safety Handbook, Kirtland AFB, NM, July 2000, available at http://www.system-safety.org/Documents/AF_System-Safety-HNDBK.pdf.

[11] National Aeronautics and Space Administration (NASA), "NASA Software Safety Guidebook", NASA-GB-8719.13, Washington, DC, March 31, 2004, available at http://www.hq.nasa.gov/office/codeq/doctree/871913.pdf.

[12] Task Force for Safety Critical Software, "Licensing of Safety Critical Software for Nuclear Reactors: Common Position of Seven European Nuclear Regulators and Authorised Technical Support Organizations," Revision 2013, available at http://www.hse.gov.uk/nuclear/software.pdf.

[13] Garrett, C., and G. Apostolakis, "Context in the Risk Assessment of Digital Systems," Risk Analysis 19(1):23–32, February 1999.

[14] Vesely, W.E., et al, "Fault Tree Handbook," NUREG-0492, January 1981, Agencywide Documents Access and Management System (ADAMS) Accession No. ML12167A103.

[15] NASA, "Fault Tree Handbook with Aerospace Applications," Version 1.1, Washington, DC, August 2002, available at http://www.hq.nasa.gov/office/codeq/doctree/fthb.pdf.

[16] SAE International, "Potential Failure Mode and Effects Analysis in Design (Design FMEA), Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes (Process FMEA)," SAE J1739, Warrendale, PA, January 15, 2009, available at http://standards.sae.org/j1739_200901/.

[17] 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, Washington, DC, November 2011, available at http://rsdo.gsfc.nasa.gov/documents/Rapid-III-Documents/MAR-Reference/GSFC-FAP-322-208-FMEA-Draft.pdf.

[18] Perrow, C., *Normal Accidents: Living with High-Risk Technologies*, New York: Basic Books, 1984.

[19] NRC, "Official Transcript of Proceedings, Nuclear Regulatory Commission: Advisory Committee on Reactor Safeguards 591st Meeting, Rockville, Maryland, Friday, February 10, 2012," ADAMS Accession No. ML12054A637.

[20] NRC, "Software-Related Uncertainties in the Assurance of Digital Safety Systems—Expert Clinic Findings, Part 1", Research Information Letter (RIL)-1001, May 2011, ADAMS Accession No. ML111240017.

[21] International Organization for Standardization (ISO), "Road vehicles—Functional safety—Part 2: Management of functional safety", Draft International Standard (ISO/DIS) 26262-2, July 2009.

[22] ISO and International Electrotechnical Commission (IEC), "Software engineering—Software product Quality Requirements and Evaluation (SQuaRE)—Guide to SQuaRE," ISO/IEC 25000:2005, Geneva, Switzerland, 2005.

[23] U.S. Federal Aviation Administration, *System Safety Handbook*, Washington, DC, December 30, 2000, from http://www.faa.gov/regulations_policies/handbooks_manuals/aviation/risk_management/ss_handbook/.

[24] Miller, S.P., et al., "Proving the shalls: Early validation of requirements through formal methods," *International Journal on Software Tools for Technology Transfer* 8(4/5):303–319, August 2006.

[25] Miller, S.P., et al., "Software Model Checking Takes Off," *Communications of the ACM* 53(2):58–64, February 2010.

[26] International Atomic Energy Agency, "Defence in Depth in Nuclear Safety," International Nuclear Safety Advisory Group (INSAG)-10, Vienna, Austria, 1996.

[27] Garrett, C., and G. Apostolakis, "Automated hazard analysis of digital control systems," *Reliability Engineering & System Safety* 77(1):1–17, July 2002.

[28] NRC, "Ineffective Use of Vendor Technical Recommendations," Information Notice 2012-06 April 24, 2012, Agencywide Documents Access and Management System (ADAMS) Accession No. ML112300706.

[29] NRC, "Age-Related Capacitor Degradation," Information Notice 2012-11, July 23, 2012, Agencywide Documents Access and Management System (ADAMS) Accession No. ML120330272.

[30] U.S. Department of Homeland Security (DHS) and The MITRE Corporation ("MITRE"), "Common Vulnerabilities and Exposures" (CVE), available at http://cve.mitre.org/.

[31] DHS and MITRE, "Common Weakness Enumeration" (CWE), available at http://cwe.mitre.org/.

[32] IEC, "Nuclear power plants—Instrumentation and control systems important to safety—Software aspects for computer-based systems performing category A functions," IEC 60880:2006, Geneva, Switzerland, 2006.

[33] NRC, "Software Safety Hazard Analysis," NUREG/CR-6430, February 1996, ADAMS Public Legacy Library Accession No. 9602290270.

[34] NRC, "High Integrity Software for Nuclear Power Plants: Candidate Guidelines, Technical Basis and Research Needs; Executive Summary," NUREG/CR-6263 (MTR 94W0000114), Volume 1, June 1995, ADAMS Accession No. ML063470590.

[35] Peña, V., R.M. Whelan, and S.V. Howieson, "Best Practices for Federal Research and Development Partnership Facilities," IDA Paper P-5148, Institute for Defense Analyses, Alexandria, VA, June 2014, available at https://www.ida.org/~/media/Corporate/Files/Publications/STPIPubs/2014/ida-p-5148.ashx.

[36] Executive Office of the President of the United States, "Trustworthy Cyberspace: Strategic Plan for the Federal Cybersecurity Research and Development Program," Washington, DC, December 2011, available at http://www.whitehouse.gov/sites/default/files/microsites/ostp/fed_cybersecurity_rd_strategic_plan_2011.pdf.

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][2][3]).

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 in which 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 [4] refers to an effect that causes different signals to become indistinguishable (or *aliases* of one another) when sampled. It also refers to the distortion that results when the signal reconstructed from samples is different from the original continuous signal.

Notes:

1. Aliasing is caused when frequencies higher than one half of the sampling rate are present (by the Nyquist Theorem, the maximum reproducible frequency is one-half the sampling rate). This results in the higher frequencies being “aliased” down to look like lower frequency components. (Adapted from definition 1 for “anti-aliasing” in [5]).
2. Aliasing is prevented through lowpass filtering of the incoming signal to block out frequencies higher than those that can be accurately reproduced by the given sampling rate. This technique is called anti-aliasing. (Adapted from definition 1 for “anti-aliasing” in [5]).

Accountability

The quality or state of being accountable (responsible).

Assumption

A premise that is taken for granted, i.e., not validated. Often, It is taken for granted implicitly

Notes:

1. This definition is used in the context of reasoning as a part of safety analysis.
2. Other forms: Assume. Assumed. Assuming.
3. In the course of engineering, a premise may be validated. Then, it is not an assumption anymore.
4. If the premise is not validated, engineering may progress by mapping the assumption explicitly into a limitation on the use of the system or a condition of use or constraint on usage. Then, the constraint is treated as any other safety requirement (e.g., hazard analysis evaluates that satisfaction of the condition is verifiable. Verification activities verify that the condition is satisfied).

Analysis

A process of reasoning showing that a proposition can be deduced from premises (adapted from [6]).

Notes:

1. The process may entail decomposition. http://plato.stanford.edu/entries/analysis/s1.html#KD.
2. See Kant's discussion at http://plato.stanford.edu/entries/analysis/s1.html#Kant.
3. Analysis may take various forms:
   - 3.1. Quantitative
     - 3.1.1. Numerical (e.g., analysis of a continuous control algorithm)
     - 3.1.2. Logical
     - 3.1.3. Other forms of mathematical analysis; i.e., where:
       - 3.1.3.1. The reasoning is composed with clear mathematical rules of composition.
       - 3.1.3.2. The reasoning is backed by science (e.g., cause-and-effect laws of engineering).
   - 3.2. Qualitative[98], but consistently[99] repeatable by comparably qualified performers.
4. Performance of the analysis may entail various degrees of machine assistance:
   - 4.1. Complete mechanization
   - 4.2. Mechanization requiring manual intervention; e.g., human-guided machine processing.
   - 4.3. Completely manual, but consistently repeatable by comparably qualified performers.
5. The term "formal" (along with its variations) is used to mean "mathematical" as in note 3.1.3.
6. Derived forms:
   - 6.1. Analyzability
   - 6.2. Analyzable
   - 6.3. Unanalyzable
   - 6.4. 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 [7]), where:

1. "Externally visible properties" of an element include behavior (both normal and abnormal) as seen from outside the boundary (interface) of an element.
2. "Relationships" include interactions and interconnections (communication paths).

Assure

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

Notes:

1. For example, by proof; see note 3.1.3 in Analysis.
2. Examples of claims:
   - 2.1. The system is safe. (Property: Safety. Value: "Is safe.")
   - 2.2. Property X of the system holds.

---

[98] See Quality.

[99] If the analysis is not consistently repeatable or the analysis method/tool itself is not qualified for safe use, the system is considered unanalyzable for the purposes of this RIL.

3. Derived forms:
   3.1. Assurance: Certainty of something (Entry 2.1 for Assurance in [1]
   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 [8].)

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 [9].

Notes:

1. Arbitrary behavior of an element that results in disruption of the intended system behavior.
2. An element of a system may exhibit a type of behavior, in which it sends conflicting information to different receivers in the system.
3. Different observers see different states, because the sender sent them different information.
4. Different observers see different states, because they access changing information at different times (e.g. reading a clock while it is changing).
5. Byzantine fault: a fault presenting different symptoms to different observers.
6. Byzantine failure: the loss of a system service due to a Byzantine fault in systems that .require consensus.

Claim

A true-false statement about the value of a defined property of a system. (Adapted from [10].)

Notes:

1. A "property" is a quality attribute of the system. (Adapted from 4.3.9 and 4.4.1 in [11].)
   1.1. Example of a property: Safety.
2. A property may have supporting sub-characteristics [11].
   2.1. Example: Verifiability ← Analyzability ← "Freedom from interference".
3. Unlike physical quantities, a property's sub-characteristic might not be measurable on an absolute scale [11].
   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 [11].
   4.1. Example sub-characteristic of the 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 [11].
6. "Value" may be invariant, may depend on time, or may depend on some other conditions [11].
7. A duration of applicability may be associated with a property (i.e., the property might not be limited to the present). For example, the property may concern the future behavior of the system [10].

8. Uncertainty (lack of certainty) may be associated with the property [10].
   8.1. The value of uncertainty might not necessarily depend on 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 on certain conditions that are formulated in terms of assumptions that the identified uncertainties do not exist.

Cognitive process

The performance of some composite cognitive activity; an operation that affects mental contents.

Collective mindfulness

A characteristic of an organization of having the collective mindset necessary to detect and understand unanticipated conditions[100] and to recover from them before they lead to harm.

> 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[101] that needs to be corrected and new sources of ignorance that become new imperatives for noticing.

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 [5].)

Notes:

1. The selection[102] of this definition was favored by Dr. Gerard Holzmann [12].
2. The term "simplicity," the converse of complexity, is often used to discuss the same issues. It is defined in [5] as follows: The degree to which a system or component has a design and implementation that is straightforward and easy to understand.
3. A "complexity measure or indicator" is distinct from the concept of "complexity."
   3.1. See definition (1)(B) in [5] for usage as complexity measure.
   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 [5].
   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) [13]. 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., the computation time is 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

[100] In the context of RIL-1101, these are mapped into "(contributory) hazards."

[101] In the context of RIL-1101, deviation is mapped into "(contributory) hazard."

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

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 [14]. 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[103] 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 [15] 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 to be integrated with reasoning to justify the completeness of coverage provided (summarized as development assurance). Examples include the following:
   2.1. Evaluation of the system concept (and conceptual architecture).
   2.2. Evaluation of the verification and validation plan.
   2.3. Criticality analysis.
   2.4. Evaluation of the architecture, including requirements allocation.
   2.5. Evaluation of the hazard analysis internal to the system.
   2.6. Validation of requirements and constraints on the design and implementation.
   2.7. Assessment and audit of all processes, including supporting and management processes.
   2.8. Certification[104] of organizations developing software.
   2.9. Evaluation of the independence[105] of the assurance activities.
   2.10. See [15] 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 assurance of the languages and tools.

Comprehensibility

The extent to which the information is easy to understand and valid inferences can be drawn from it (adapted from definition of comprehensible in [1]).

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 [16]).

---

[103] This refers to behaviour under all foreseeable operating conditions.

[104] 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.

[105] For example, independence can be evaluated through certification of the assurance process for the complex logic (e.g., software).

Examples:

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

Contribute

Help to cause or bring about something (Definition 1.1 in [1]).

Notes:

1. Derived forms:
   1.1. Contribution: The thing contributed.
   1.2. Contributory: Of, relating to, or forming a contribution.
2. 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” allows their inclusion.
3. Definition of cause in [1]: Make something (typically something bad) happen (occur).

Contributory hazard

Factor contributing to potential for harm.

Notes:

1. (Excerpt from [17]:) “…. An unsafe act and / or unsafe condition which contributes to the accident[106] ....”.
2. Figures 7-1 through 7-4 in [18] illustrate contribution paths.
3. Examples:
   3.1. The potential for adverse energy flow [17].
   3.2. Inappropriate functions (from Figure 7-5 in [18]).
   3.3. Normal functions that are out of sequence (from Figure 7-5 in [18]).
   3.4. Functional damage and system degradation (from Section 7.1.1 in [18]).
   3.5. Machine-environment interactions resulting from change or deviation stresses as they occur in time and space (from Section 7.1.1 in [18]).

Cultivate

Develop (improve) a pattern of behavior.

Data

Value, symbol, image or other representation in a form which can be communicated across its users.

Notes:

1. Data is used as a singular noun, as well as, plural.
2. The users may be people or machines.
3. Data on its own has no meaning. Also see Information.

Design Defect

Frailty or shortcoming of an item resulting from a defect in its concept, and which can be avoided only through a redesign of the item. (Adapted from [19])

Notes:

1. In RIL-1101, the term is used primarily in the context of the engineering phases of the product lifecycle.

[106] Or, for the purpose of this document, to the degradation of a safety function.

2. Definition 2 in 3.764 [16] defines defect as follows, “An imperfection or deficiency in a project component which causes that component to fail to meet its requirements or specifications so that the component needs to be either repaired or replaced.” In this definition and other similar definitions of “defect”, the expression “… fail to meet its requirements or specifications” excludes cases in which the requirement or specification itself is deficient. However, most defects occur, because the requirements are deficient (e.g., incomplete, inconsistent, ambiguous, or even incorrect). According to these definitions, a system might not be defective, yet it might lead to a hazard.
3. Referring to the definition for design defect, “a defect … which can be avoided only through a redesign of the item,” if the redesign is also an attempt to satisfy the same deficient requirements, then the associated (contributory) hazard might still be present.

Demonstrate

Prove (the assertion in context) through reasoning.

Notes:

1. The assertion may be a claim.
2. The premise in the reasoning may be some evidence supporting the claim.

Dependent

Determined or conditioned by another.

Notes:

1. Other forms:
   1.1. Dependency: The quality or state of being dependent on or unduly subject to the influence of another.
   1.2. Dependence: Same meaning as dependency.
   1.3. Independent
2. For example, if some factor or condition could cause the degradation of a safety function, then the safety function is dependent on it. Also see “Contributory.” A safety function could be dependent on a contributor in many ways (paths or channels or couplings) In addition to direct causal paths, a dependency could arise through a side effect such as interference across activities and resources.

Deterministic

(In the context of a process) such that the resulting behavior is entirely determined by the initial state and inputs to the process, and which is not random or stochastic.

Notes:

1. The terms “deterministic” and “predictable” are related as follows.
2. Predictability: The degree to which a future state of the system can be determined, i.e., known, given the current state and for a given set of inputs.
   2.1. For a logic system, the “as built” system should behave exactly as predicted through the analysis that is used in its assurance.
   2.2. For a physical (e.g., electrical, hydraulic, or mechanical) system, the “as built” system should behave as predicted, within specified limits, which are used in the analysis for its assurance. Sometimes, these limits are known as error bounds. The behavior is considered deterministic, because the variations are attributed to causal factors which were not modeled with sufficient accuracy in the analysis. Example causes for loss of accuracy: Discretization of continuous phenomena, such as value of pressure and time of its measurement; inaccuracy of measuring instrument; coulomb energy losses; coulomb friction; variation in geometry of a physical object.

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 [20]).

Notes:

1. The term “discrete constituent” is substituted for the word “component” used in the definition from [20] to 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, an interface with its environment in accordance with the definition in [21]) and an intrinsic, immutable, unique identity (adapted from [20]).
2. In general, an element is a discrete part of a system that can be implemented to fulfill specified requirements.
3. Examples:
   3.1. Hardware element.
   3.2. Software element.
   3.3. Human element.
   3.4. Data element.
   3.5. Data structure.
   3.6. Process (e.g., a process for providing service to users).
   3.7. Procedure (e.g., operating instructions).
   3.8. Facility.
   3.9. Material.
   3.10. Naturally occurring entity (e.g., water, an organism, or a mineral).
   3.11. Any combination of these things.
4. An element may have other elements in it (e.g., a subsystem).
5. 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. (A combination of 9A and 9B in [5]).

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
3. The “environment” of a 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 corresponding manual procedures.

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 [5].)

Evidence

Data supporting the existence or truth of something. (Adapted from 3.1936 in [16].)

Notes:

1. Examples of means of obtaining “raw” evidence: Test, measurement, and 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 a claim for the next higher level of integration of a system.

Failure

The termination of the ability of an item to perform a required function. [22]

Notes:

1. After failure, the item has a fault. [22]
2. “Failure” is an event, as distinguished from “fault,” which is a state. [22]
3. This concept as defined does not apply to items consisting of software only. [22]
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 [5]).
   4.2. The termination of the ability of a functional unit to perform its required function (Definition (1)(N) in [5]).
   4.3. An event in which a system or system component does not perform a required function within specified limits; a failure might be produced when a fault is encountered (Definition (1)(O) in [5]).
   4.4. The termination of the ability of an item to perform its required function (Definition 9 in [5]; from a former standard for “nuclear power generating station”).
   4.5. The loss of ability of a component, equipment, or system to perform a required function (Definition 13 in [5]).
   4.6. An event that might limit the capability of equipment or a system to perform its function(s) (Definition 14 in [5] from “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 [5] from a former standard for “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 [22].)

Failure mode

The effect by which a failure is observed to occur. [23][24]

Notes:

1. A failure mode is usually characterized by description of the manner in which a failure occurs. For example, in the case of a relay which fails to close on command, the failure mode is fails to open or fails to close. [23]
2. A failure mode is not characterized in terms of the failure mechanism [23]
3. A failure mode is not characterized in terms of the failure effect. [23]
4. A set of failure modes is characterized in the context of a particular level of assembly or integration. For example, a failure mode at one level of assembly might be an effect at the next higher level of assembly. [23]
5. Referring to note 3 for failure, in RIL:-1101. the term,.failure mode, is not applied to a software item.

Failure modes and effects analysis (FMEA)

A qualitative method of reliability analysis which involves the study of the failure modes which can exist in every subitem of an item, as well as the determination of the effects of each failure mode on other subitems of the item and on the required functions of the item. (191-16-03 in [22].)

Note:

1. Referring to note 3 for failure and note 5 for failure mode, in RIL:-1101, the term, FMEA, is not applied to a software item.

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 because of lack of external resources. (191-05-01 in [22])

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.
5. Latent fault: Fault remaining in the digital safety system placed in operation. Also see "resilience."

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 [22])

Fault Mode

One of the possible states of a faulty item, for a given required function. (191-05-22 in [22])

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 [22])

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 [16]).

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 [16]).
2. Derived form: "Fault tolerant" or "fault-tolerant": Pertaining to a system or component that is able to continue normal operation despite the presence of faults (3.1128 in [16]).
3. For example: Conditions that might 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, might result in a stated fault mode of the item, presented in the form of a fault tree (191-16-05 in [22]).

Feasible

Capable of being done with the means at hand and circumstances as they are. (Entry for "feasible" in [3])

Notes:

1. Other definitions also impose such constraints as:
   1.1. Practicability.
   1.2. Reasonable amount of effort, cost, or other hardship. [25]
   1.3. Ease and convenience. (Entry for feasible in [1]).
2. Such constraints distinguish "feasibility" from "possibility."

Freedom from interference

Freedom from degradation of the performance of a function resulting from interaction across the system and its environment or interaction 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 [5]).

Note: The interconnections referred to here are at the level of the printed circuit board or cabinet, not those internal to integrated circuits.

Hazard

Potential for harm.[109]

Notes:

1. Usage of the term, hazard, in RIL-1101 is bounded to the context of the object of analysis (e.g. a digital safety system for an NPP) and its defined environment. Usage of the term, hazard, without such a context is not meaningful.
2. Definition A in [26] (which is the same as definition 3.1283-1 in [16]) elaborates on the "potential for harm" as follows, "An intrinsic property or condition that has the potential to cause harm or damage."
3. At the initial stage of hazard logging (before any analysis of the initial finding), the log might include an item, which is identified as a hazard, but, after some analysis, is recognized not to be a hazard as elaborated in note 2 (e.g., it might be recognized as an event).and recharacterized.
4. Examples:
   1.1. A potentially harmful condition.
   1.2. A potentially harmful circumstance.
   1.3. A potentially harmful scenario.

[107] Examples: Wiring in cabinets and printed paths in circuit boards.

[108] Examples: Relays, AND-gates, and OR-gates.

[109] In general, potential for "loss" of any kind that is of concern, but the focus of RIL-1101 is potential for harm through the degradation of a safety function allocated to the object under analysis.

Hazard analysis

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

Notes:

1. The "hazard identification" part of HA includes the identification of losses (harm) of concern.
2. This definition is narrower than many definitions of HA, as explained below:
   1.1. The scope of the definition excludes the verification that the requirements and constraints have been satisfied.
   1.2. The scope of HA is limited to identification of hazards (including contributors) and formulation of corresponding constraints. Activities to satisfy these constraints (e.g., architectural design, detailed design, implementation, and associated processes) are treated as part of the development process.
   1.3. The scope of the definition does not explicitly include quantification. 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 of defining its characteristics [16].

Indicate

To be a sign or symptom of, (Indicate in [1]).

Notes:

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. [5]
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 in facilitating comparison of the current state with the goal state rather than for absolute accuracy.
4. Contrast with quality measure.

Information

Data that has been given some meaning within a particular context, such that it can be shared among its users. (Adapted from 3.1396 in [16]).

Intended

Intentional. (Meaning 2 in [2].)

Notes:

1. An intended item might be one that is not a direct, explicit requirement, but could have been derived from an explicit goal or requirement.
2. Derived form: "Unintended," meaning "not intentional" (i.e., it was not even required indirectly or implicitly).
3. Also see unwanted.

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).

Interaction

A kind of action that occurs when an object affects another.

Notes:

1. An interaction may affect more than one object.
2. An interaction may involve more than two objects.
3. One object may affect another through other intermediate objects (i.e., the interaction might be indirect, implicit, or a side effect of some other interaction).
4. The direct effect of an interaction might not be observable, when the system or its environment are in a steady state.
5. An interacting object in the environment of the affected object may be a human, an automated system, or data.
6. The effect of an interaction may be time-delayed.
7. The number of intermediaries and delays might obscure the cause-effect relationship.
8. An interaction might be unintended, unwanted or unspecified.
9. An interaction might result from some abnormality; for example, invalid input, a hardware malfunction, or a human mistake.

Item (entity)

Any part, component, device, subsystem, functional unit, equipment, or system that can be individually considered. (191-01-01 in [22])

Notes:

1. In [26], 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

Introduce machines or automatic devices into a process or activity (Entry for mechanize in [1]).

Mistake

A human action that produces an incorrect result (Definition 3 in [5]).

Notes:

1. In the context of developing a software-dependent system, this definition is applicable to mistakes concerning requirements development; for example:
   1.1. Elicitation.
   1.2. Transformation of intent into requirement or constraint specification.
   1.3. Explicit statement of assumptions (e.g., about the environment).
   1.4. Respective V&V activities.
2. Similarly, this definition is applicable to hazard analysis activities, which are critically dependent upon the performer’s competence. See Section C.4 in Appendix C.

3. 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). [5]

Mode

A subset of all the possible functionality and behaviors of a system.

Notes:

1. See Appendix H for examples of NPP modes.
2. Referring to the example in Appendix H, the collection of all the modes would characterize all the functionality and behaviors of an NPP-level system.
3. In a well-engineered system, valid or eligifble transitions across modes are well defined.
4. The concept of modes and mode transitions can also be applied to finer levels of integration or assembly.
5. The concept of mode is similar to the concept of state, but the difference is that a state characterizes the exact operating condition of a system.

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 mode or state.

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.

Process

A set of interrelated activities which transforms inputs into outputs. (Definition 12(A) in [5] and Definition 3.2217-1 in [16].)

Notes:

1. Definition 4 in [5] makes "including the transition criteria for progressing from one (activity) to the next" explicit.
2. In definition 4 in [5], the expression "that brings 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, and implementation. If the overall engineering is considered a lifecycle process, these may be identified as phases in that lifecycle process.

Product

Result of a process. (3.2257-4 in [16])

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.
2. System-requirements specifications, system-architecture specifications, detailed design specifications, (software) source code, and (software) executable code can all be considered "products."

Quality

Capability of product to satisfy stated and implied needs when used under specified conditions. (Adapted from 4.51 in [27].)

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 might not have been stated explicitly (e.g., a need that is considered to be evident or obvious or a need implied by another stated need)."
3. In this context, the term "quality of service" has also been used to mean "quality."
4. A "quality model" is a defined set of characteristics, and of relationships between the characteristics, which provides a framework for specifying quality requirements and evaluating quality. (Adapted from 4.44 in [27].)
5. A "quality measure" is an attribute of quality to which a value is assigned. Also see scale.
6. "Quality in use" is the capability or property 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.
    6.1. Actual quality in use might be different from quality in use measured in a test environment earlier in the product lifecycle, because the actual needs of users might not be the same as those reflected in the test cases or in the requirements specifications.
    6.2. Quality-in-use requirements contribute to identification and definition of external software quality requirements.
    6.3. An example of quality in use: Safety (freedom from harm).
7. "Measurement of external quality" refers to measurement from an external view of the product, in which 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 -se requirements, such as safety.
8. "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 entry for argument in [2]).

Notes:

1. "Argument": Also see [28].
2. Derived forms:
    2.1. Reasoning: The use of reason.
    2.2. Reasonable: Being in accordance with reason (entry for reasonable in [2]).

Reliability (symbol: $R(t_1, t_2)$) where t_1 and t_2 are the start and end times of the interval respectively.

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 [22]).

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.[110]
2. The term "reliability" is also used to denote the reliability performance quantified by this probability (see 191-02-06 in [22]).
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 for something to be accomplished or realized (adapted from 4.47 in [27]).

Notes:

1. A "functional requirement" is a requirement that specifies a function that a system or its element must be able to perform (adapted from 4.22 in [27]).
2. A "quality requirement"[111] is a 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.
   2.2. External quality. Also see note 7 in the definition of quality.
   2.3. Internal quality. Also see note 8 in the definition of quality.

Resilience

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

Notes:

1. "Resilience" has been used to describe a property of a system; however, this meaning 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 and 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. (Entry for resilience in [3])

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 [16].)

Scale (for a quality measure)

An ordered set of values, continuous or discrete, or a set of categories to which an attribute is mapped (adapted from 2.35 in [8]).

---

[110] 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.

[111] Colloquially, these are also known as non-functional requirements.

Notes:

1. The type of scale depends on the nature of the relationship between the values on the scale [8].
2. Four types of scale are commonly defined [8]:
   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 [29].

State

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

Notes:

1. The condition represented by a state is an abstraction of the complete set of observable properties (also known as state variables) that characterize the behavior of a system.
2. The behavior of a system is characterized as its response to stimuli (set of inputs). The response may result in some output and state-change (i.e., change in the set of values of its state variables).

State space

The set of all possible states of a dynamic system [30].

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 [31].

Notes:

1. A system may be considered as a product or as the services it provides (adapted from [31]). 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 words “combination” and “organized” (instead of “collection”) emphasize that a system is an “integrated composite” as characterized in the definition for “system” in [32].
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 for “system” in [32].
4. In practice, the interpretation of its meaning is frequently clarified by the use of an associated noun or nouns (e.g., reactor-protection system). (Adapted from [31].)
5. RIL-1101 is focused on a digital safety system and its interactions with its environment. Operators, thermo-hydraulic processes, and related supporting peripheral processes are part of the environment.
6. At the concept phase of the system lifecycle, a system may have no identified internal elements.
   6.1. Then, it may be characterized, studied, or analyzed in terms of its behavior and its interactions with its environment.
   6.2. In the trivial case, the system may have no identified constituent elements.

7. Systems can be composed of systems. A system with only software elements is also a system. For example, if a program is composed of subroutines, the subroutines are elements and the program is a system.

Systemic

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

Note:

1. For example, organizational culture and competence are systemic causes, which can affect more than one element in a system in more than one way.

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 [22].

Note:

1. Examples of causes of systematic failures include human mistakes in the following activities:
   1.1. The system safety requirements and constraints.
   1.2. The specification, design, manufacture, , or integration, or configuration of the hardware.
   1.3. The specification, design, implementation, or integration, or configuration of the software.

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 [7].)

Note:

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 [16].)

Notes:

1. “Confirmation” is used instead of “assurance,” the word used in [16], for these reasons:
   1.1. To avoid confusion with the use of the word “assurance” in RIL-1101.
   1.2. To achieve consistency with the use of “confirmation” in the definition of “verification.”
   1.3. “Confirmation” subsumes the term “the process of evaluating” used in definition A in [26].
   1.4. “Confirmation” subsumes the term “the process of providing evidence” used in definition B in [26].
2. “Validation” includes confirmation that the requirements are correct, complete, consistent, and unambiguous.
3. The stakeholder-requirements definition activity includes the transformation of various needs into requirements, including the requirements for validation [15].
   3.1. In [26], validation of the stakeholder-requirements definition includes HA.
   3.2. In the context of an NPP safety system, “stakeholder requirements” mean NPP safety requirements allocated to and intended for this safety system.
   3.3. “Requirements for validation” include assurability.
4. The activity of validation includes the confirmation that the specification for each lifecycle phase satisfies the needs of the customer and other identified stakeholders.

5. A clarification of the expression "the needs of the customer and other identified stakeholders" is provided in definition B in [26] 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.
6. 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."
7. "Product" subsumes the elaboration "system, software, or hardware and its associated products" used in definition B in [26].
8. "Satisfies" is used instead of "meets," the word used in [16] in order to maintain consistency with the usage in the definition of "verification."
9. The elaboration "….satisfy requirements allocated to it at the end of each life cycle activity" in definition B in [26] 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 [16].)

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, "confirmation" and "process of confirmation."
    2.1. Definition A in [26] defines verification as "The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase." The act of evaluating includes reviewing, inspecting, testing, checking, auditing, or otherwise determining and documenting (also see note 9 below). The term "confirmation" in the definition is used to mean that the result of the determination is TRUE."
    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 [16] uses the term "fulfilled"; however, to reduce potential ambiguity, the term "satisfied" is used (which is also used in definition 1 in [16]) in the general sense of propositional satisfaction and constraint satisfaction.
    3.1. Definition 2 in [16] uses the term "formal proof," favoring this substitution.
    3.2. Definition 6 in [16] uses the term "comply with," which may be mapped conservatively into "satisfy."
    3.3. Definition B in [26] uses the term "conforms to," which may be mapped conservatively into "satisfies."
4. Definitions 3 and 6 in [16] 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 [26] 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 [26] elaborates "… for all life cycle activities during each life cycle process"; the definitions of product and process subsume this elaboration.
7. Definition B in [26] 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 the definitions of product and process subsumes this elaboration.
8. Definition B in [26] 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 [5] 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 [26] 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] Oxford University Press, available at http://www.oxforddictionaries.com/us/definition/american_english/.

[2] Merriam-Webster, Incorporated, "Definition and More from the Free Merriam-Webster Dictionary," available at http://www.merriam-webster.com/dictionary/.

[3] Princeton University, "WordNet Search - 3.1" available at http://wordnetweb.princeton.edu/perl/webwn?.

[4] Wikipedia.org, "Aliasing - Wikipedia, the free encyclopedia," available at http://en.wikipedia.org/wiki/Aliasing.

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

[6] Caygill, H., *A Kant Dictionary*, Hoboken, NJ: Wiley-Blackwell, July 1995; for a summary of its definition of "analysis," see http://plato.stanford.edu/entries/analysis/s1.html#KD.

[7] Bass, L., P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd edition, Boston: Addision-Wesley, April 2003, as quoted at http://www.sei.cmu.edu/architecture/start/glossary/moderndefs.cfm.

[8] ISO and IEC, "Systems and software engineering—Measurement process," ISO/IEC 15939:2007(E), Geneva, Switzerland, 2007.

[9] Schneider, F.B., "Understanding Protocols for Byzantine Clock Synchronization," Cornell University, Ithaca, New York, Technical Report (TR)-87-859, August 24, 1987, available at http://ecommons.library.cornell.edu/bitstream/1813/6699/1/87-859.pdf.

[10] ISO and IEC, "Systems and software engineering—Systems and software assurance—Part 1: Concepts and vocabulary," ISO/IEC DIS 15026-1:2013, Geneva, Switzerland, 2013.

[11] ISO and IEC, "Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—System and software quality models," ISO/IEC 25010:2011, Geneva, Switzerland, 2011.

[12] U.S. Nuclear Regulatory Commission, "Software-Related Uncertainties in the Assurance of Digital Safety Systems—Expert Clinic Findings, Part 1", Research Information Letter (RIL)-1001, January 2011, Agencywide Documents Access and Management System (ADAMS) Accession No. ML111240017.

[13] Goldreich, O., *Computational Complexity: A Conceptual Perspective*, Cambridge, UK: Cambridge University Press, April 2008.

[14] Cobham, A., "The intrinsic computational difficulty of functions," in Bar-Hillel, Y., ed., *Logic, Methodology, and Philosophy of Science, Proceedings of the 1964 International Congress*, Amsterdam: North-Holland Publishing, 1965.

[15] RTCA, Inc., and the European Organisation for Civil Aviation Equipment (EUROCAE), "Design Assurance Guidance for Airborne Electronic Hardware," RTCA DO-254/EUROCAE ED-80, Washington, DC, and Malakoff, France, April 19, 2000.

[16] ISO, IEC, and IEEE, “Systems and software engineering—Vocabulary,” ISO/IEC/IEEE 24765:2010, Piscataway, NJ, 2010.

[17] AviationGlossary.com, “Contributory Hazard | Aviation Glossary,” available at http://aviationglossary.com/aviation-safety-terms/contributory-hazard/.

[18] Federal Aviation Administration, “FAA System Safety Handbook, Chapter 7: Integrated System Hazard Analysis,” Washington, DC, December 30, 2000, available at http://www.faa.gov/regulations_policies/handbooks_manuals/aviation/risk_management/ss_handbook/media/Chap7_1200.pdf.

[19] BusinessDictionary.com, “What is design defect? definition and meaning,” available at http://www.businessdictionary.com/definition/design-defect.html#ixzz3H4G4twDt.

[20] Institute of Electrical and Electronics Engineers, IEEE Standard 1233-1998, “IEEE Guide for Developing System Requirements Specifications,” Piscataway, NJ, 2012, 1998.

[21] ISO and IEC, “Information Technology—Vocabulary—Part 1: Fundamental Terms,” ISO/IEC 2382-1:1993, Geneva, Switzerland, 1993.

[22] IEC, “International Electrotechnical Vocabulary, Chapter 191: Dependability and Quality of Service,” IEC 60050-191:1990-12, Edition 1.0, Geneva, Switzerland, 1990.

[23] Institute of Electrical and Electronics Engineers, IEEE Standard 500-1984 P&V, “IEEE Standard Reliability data for Pumps and Drivers, Valve Actuators, and Valves,” excerpted from ANSI/IEEE Std 500-1984, New York, NY, 1986.

[24] Institute of Electrical and Electronics Engineers, IEEE Standard 1100-2005, “IEEE Recommended practice for Powering and Grounding Electronic Equipment,” Piscataway, NJ, 2005.

[25] U.S. Department of Transportation, Federal Highway Administration, “Appendix B: Glossary - Sidewalks - Publications - Bicycle & Pedestrian Program - Environment - FHWA,” available at http://www.fhwa.dot.gov/environment/bicycle_pedestrian/publications/sidewalks/appb.cfm.

[26] Institute of Electrical and Electronics Engineers, IEEE Standard 1012-2012, “IEEE Standard for System and Software Verification and Validation,” Piscataway, NJ, 2012.

[27] ISO and IEC, “Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—Guide to SQuaRE,” ISO/IEC 25000:2014(E), Geneva, Switzerland, 2005.

[28] Toulmin, S, The Uses of Argument, Cambridge, UK: Cambridge University Press, 1958.

[29] Wikipedia.org, “Separation of concerns - Wikipedia, the free encyclopedia,” available at http://en.wikipedia.org/wiki/Separation_of_concerns.

[30] Scholarpedia.org, “State space - Scholarpedia,” available at http://www.scholarpedia.org/article/State_space.

[31] IEEE, “Systems and Software Engineering—Software Life Cycle Processes,” IEEE Standard 12207-2008, Piscataway, NJ, 2008.

[32] U.S. Department of Defense, “Department of Defense Standard Practice: System Safety,” MIL-STD-882E, Washington, DC, May 11, 2012, available at http://www.system-safety.org/Documents/MIL-STD-882E.pdf.

APPENDIX B: Technical Review Process

Technical reviews of this document were performed iteratively for 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, and the 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 of material.

Review process

The technical reviews were performed iteratively at evolving stages of Research Information Letter (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. The NRC and SEC provided orientation to the expert as follows
   1.1. The 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; and
      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, the 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 how well it matched the 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., by generalization through inductive or abductive reasoning).
3. NRC staff and the expert discussed the expert's responses in a teleconference moderated by SEC. Most of the responses concerned the 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. The NRC proposed disposition of expert's suggestions, sometimes including followup questions for discussion with the expert.
6. The NRC discussed its proposed disposition with the expert. Depending on need and scheduling feasibility, sometimes the 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 between experts about technical soundness.

Reference for Appendix B

[1] U.S. Nuclear Regulatory Commission, "Digital Instrumentation and Control - Technical Engineering Services," commercial contract V6065, September 2012, Agencywide Documents Access Management System (ADAMS) Accession No. ML12284A214.

APPENDIX C: Evaluating Hazard Analysis—State Of The Art

The scope of this appendix is limited to the scope of Research Information Letter (RIL)-1101, especially analysis of contributory[112] hazards in digital safety systems for nuclear power plants (NPPs), which are rooted in systemic causes. For example, it does not elaborate on analysis of hazards from random hardware failure. It does not discuss analysis of systems with a mix of safety and non-safety functions (mixed-criticality systems). Whereas almost all of the surveyed publications are much broader in scope, this appendix maps the extracted information into the narrower scope of this RIL. For example:

1. It covers only a relevant subset of the wide range of hazard-analysis (HA) activities.
2. It does not discuss techniques exclusively suited to analysis for random failure of hardware components in a system.
3. Its starting point, the loss of concern, is the degradation of a safety function allocated to an NPP digital safety system; in contrast, the NPP-level loss of concern would be the unwanted release of radioactivity.

C.1 Contextual interpretation of terms

The specific interpretation of the terms, hazard and hazard analysis, depends upon the context. Section C.1.1 reviews the general context that was introduced in the glossary definitions of these terms. Section C.1.2 illustrates different types of objects, on which HA may be performed in the course of HA for an NPP digital safety system. Section C.1.3 introduces different contexts of an object of HA, based on its position in the dependency network influencing an NPP digital safety system.

C.1.1 General context of hazard analysis

The vocabulary in this appendix is defined in Appendix A. As defined and explained therein, a hazard is potential for harm, in the context of the digital safety system being analyzed, as well as in the context of its environment. A hazard is an intrinsic property or condition (state[113]) of the system, including its interaction with its environment." 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. An intermediate work product of the system to be analyzed or an object within the system such as:
   1.1. Concept of a digital safety system; for example, a reactor-protection system (RPS).
   1.2. One of the four identical divisions of the digital safety system; the information needed for analysis (hereafter in this list, information source) may be the system architecture).

[112] IEEE 1012-2012 [1], "IEEE Standard for System and Software Verification and Validation," introduces the notion of contributory hazards; e.g., software and hardware contributions to system hazards.

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

1.3. An element responsible for the voting logic used in the system; (information source: system architecture).
1.4. A system at a lower level of integration (information source: system architecture).
1.5. The most finely grained component in the integration hierarchy (information sources: software architecture and 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).
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 Figure 4.
4. Any other object in a path of contributory hazards (i.e., in the dependency network).

C.1.3 Analysis at different levels in the dependency network

The dependency network of the top-level system provides an organizing framework for supporting HA of objects in the dependency network. For each object, the starting point of its HA would include the following:

1. The derived requirements allocated to it.
2. Its boundary with respect to its environment
3. Its relationship to its environment
4. Associated assumptions.

If HA of different objects is occurring concurrently (e.g., analysis for impact of changes), based on assumptions about their place and relationships in the dependency network, then, for the implications of these assumptions, see the following items in this RIL:

- Table 3, H-culture-12;
- Table 5, H-ProcState-4;
- Table 9, H-SR-12 through 14;
- Table 10, H-SRE-2G2; and
- Table 14, item 1 of H-SAE-1G1 and H-SAE-7.1.

C.2 Reference lifecycle model for hazard analysis

Independent hazard analysis of a digital safety system is part of its safety-analysis activities (also see Section 1.7.8). These activities are performed by people who are organizationally independent from the mainstream development. The initial HA and verification and validation (V&V) are also performed by the mainstream development organization [2]. The independent HA is intertwined with associated development engineering activities and uses its work products, 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 during various activities in the system development lifecycle:

1. V&V and HA activities of the mainstream system development, organization;
2. Independent verification activities; or

3. Independent HA activities.

In general, the higher the quality of the upstream processes, the smaller the hazard space downstream will be and the lower the number of hazards within downstream work products will be. On the other hand, ill-controlled upstream processes could leave such a large hazard space in their work products that the downstream verification and HA are rendered 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 information in [1] applicable to an NPP digital safety system. 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 through 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 examine obscure contributors such as the following:

- Interpretations of a requirement specification;
- Flowdown of derived requirements and constraints;
- Flowdown of quality requirements;
- Validity of the process specifications and resource qualification criteria; and
- Assumptions.

To the extent that premises 1 through 7 stated above are not satisfied, the deficiency results in additional burden on the independent HA activities, requiring correspondingly additional skills and effort.

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

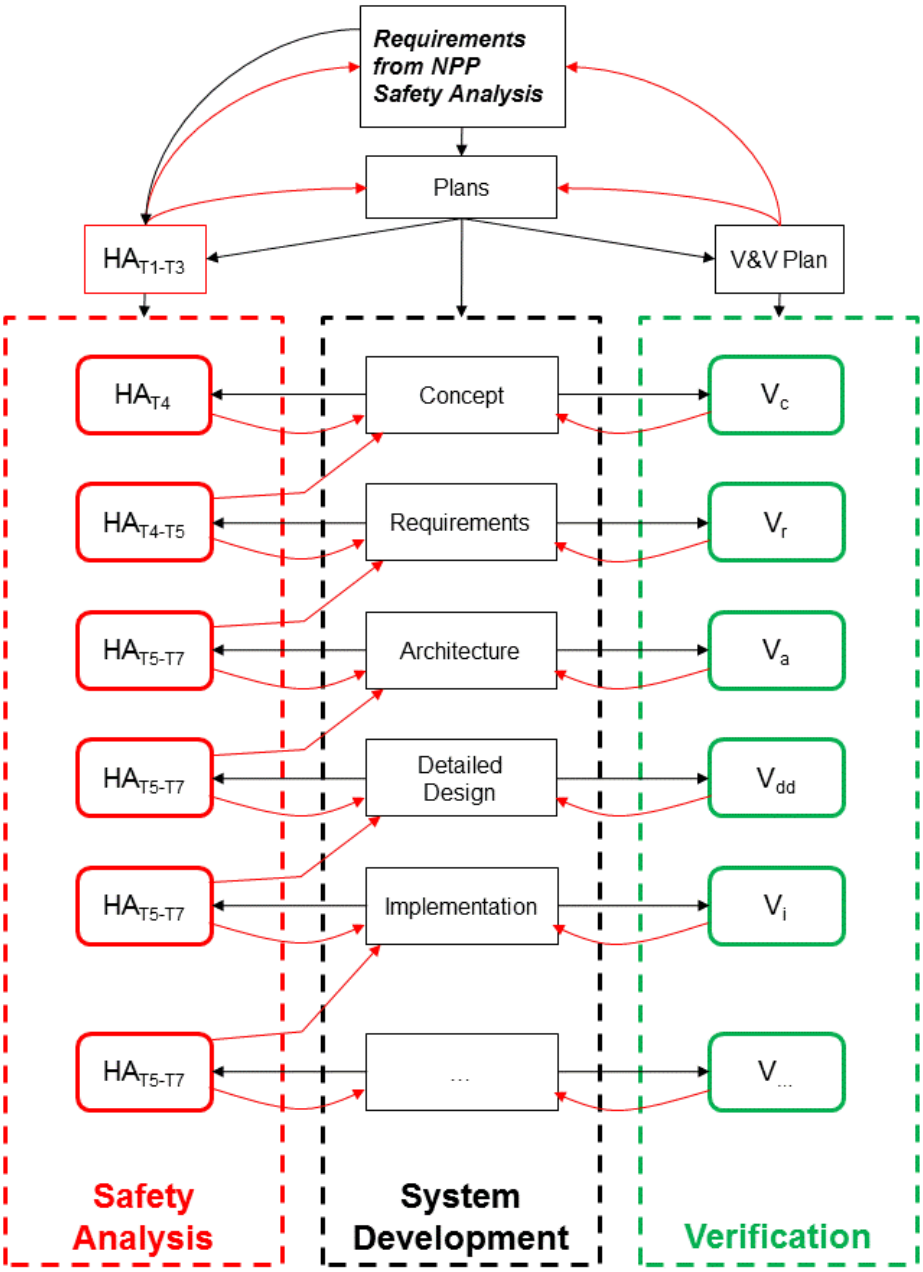

Figure 9: Hazard analysis in relation to development lifecycle and verification activities.

C.3 HA tasks—an example set

Table 21 outlines an illustrative example set of HA tasks, based on the reference model in [1]. Each of the tasks, labeled T1 through T7, is characterized in terms of a set of inputs (the information analyzed) and outputs (the results of the task). The rightmost column in the table cites the genesis of the task-formulation. An identified deficiency (e.g. inadequacy in the input information) requires some corrective action and change. Every change requires a review for the effect of the change.

Tasks T1 through T3 start in the planning phase of the system engineering lifecycle. HA in the planning phase might identify gaps in the input information and thus drive engineering effort to fill the gaps. Phase-advance clearance is the final output. Upon this clearance, the project may advance to the activities of the next phase. This clearance concept is sometimes known as a quality gate or a safety gate. Proceeding further into development without satisfying the gate criteria could result in much rework and wasted effort later. In some cases, the deficiency may be irreparable.

Task T4 is started in the concept phase of the system engineering lifecycle. In a "green-field" concept, the information available might only be a functional concept. Yet it is sufficient to develop the questions to be addressed from the HA perspective 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, traceable rationale underlying the evolution path. In a modification of an existing NPP, the concept might 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 to T5, at the start of which, the term "object" refers to the system-requirements specification (corresponding to task 203 in [3]). 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 and References.
T1. Generate baseline HA plan for all lifecycle phases.	1. Concept [1], including interactions with and dependencies on its environment. 2. Requirements from NPP level safety analysis. 3. Premises & assumptions on which the expected outcome depends, including conditions & modes of operation and maintenance. 4. Plan to validate assumptions. 5. Consequences of	Baseline[114] HA plan.	Adapted from Tasks 7.1:1 through 7.1:4 in Table 1a in [1] and Task 101.2.2 in [3].
T2. Identify dependencies of HA plan (e.g., other information, resources, and dependencies on supply chain)		Dependencies of plan.	Adapted from Tasks 7.1:1 through 7.1:4 in Table 1a in [1] and from [3].
T3. Evaluate other plans, following the dependencies identified above. Coordinate these information		1. Evaluation report. 1.1. Deficiencies. 1.2. Changes needed. 1.3. Request for additional information (RAI).	Adapted from Tasks 7.1:1 through 7.1:4, 7.4, and 7.5 in Table 1a in [1].

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

HA activity / task	Input	Output	Remarks and References.
exchanges (e.g., timing, semantic compatibility, and format) with HA activities.	behavior shortfalls, including invalid assumptions/premises. 6. Overall V&V plan, including HA. 7. Mainstream development plan. 8. Corresponding information about or from entities in the dependency paths (e.g., up the supply chain).	2. Rejection or Acceptance (including phase-advance clearance) 3. Revision to HA plan as needed.	Adapted from Tasks 1 through 4 in Table 1a in [1]. Adapted from Tasks 7.1:1 through 7.1:4 in Table 1a in [1].
T4. Understand HA-relevant characteristics of the object to be analyzed; examples: 1. Differences from previously licensed systems. 2. Exposure to unwanted interactions. 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 flowdown, decomposition, or derivation of requirements. 9. Identification of dependencies (e.g., feedback paths and hidden or obscure couplings). 10. Premises and assumptions, both explicit and implicit. 11. Other challenges to analyzability.	Items above plus: 9. Other requirements allocated to the object. 10. Nonsafety-related constraints on the object. 11. Relationship 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 and competencies), and information-exchange interfaces. 15. Identification of reused objects and conditions of use. 16. Explicit record of dependencies. 17. Prior HA results, if any.	1. Revision to HA plan. 2. Addition to hazard log. [4] 3. Change needed; examples: 3.1. Making assumptions explicit; 3.2. Improvement in knowledge of dependencies; 3.3. Making lifecycles and processes compatible; 3.4. Making information-exchange interfaces compatible; 3.5. Consistency across automation and human roles and procedures [5]. 3.6. Qualification of reused objects (e.g., tools); 3.7. Change in allocation of a requirement; 3.8. Other constraints; and 3.9. Other derived requirements. [6] 4. RAI.	Adapted from Tasks 7.2:1(a, f, and g), 7.2:2(b and d), and 7.2:3(a and b) in Table 1a in [1] and from Tasks 201 and 202 in [3].
T5. Analyze object[115] for (contributory) hazards. See the	Items above plus information specific to object of analysis	1. Addition to hazard log.	Adapted from Tasks 7.1:5

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

HA activity / task	Input	Output	Remarks and References.
corresponding section and table in RIL-1101. For a safety system or its element, it includes, for example, a search for: 1. Single-point failure; 2. Common-mode dependency; and 3. Common-cause dependency.	(see Section C.1.2).	2. Change needed. 3. Examples: 3.1. As in T4; 3.2. Derived requirement (on process) to prove that a contributing hazard cannot occur; and 3.3. Derived requirement or constraint on object. 4. Rejection or Acceptance (including phase-advance clearance). 5. Revision to HA plan as needed. 6. RAI.	and 7.1:6 in Table 1a in [1], from Table 1b in [1], and from [7].
T6. Integrate analyses from lower levels in the integration hierarchy and contribution paths up to the top-level analysis.	Items above plus information needed about inter-object dependencies for overall system HA.	As in T5.	Adapted from Task 7.1:7 in Table 1a in [1] and from other portions of [1].
T7. Analyze change proposal (e.g., hazard-control proposal).	The change proposal, including information on which it depends (e.g., items listed above).	As in T5.	Abstracted from [1].

Referring to Sections C.1.2 and C.1.3, as the analysis identifies dependencies and the objects on which the findings are dependent, relevant tasks in Table 21 are performed on each identified object. When multiple inter-dependent objects are evolving concurrently, HA on these objects may be performed concurrently, formulating the needed assumptions about the inter-dependencies. These assumptions may be used as constraints on the inter-dependencies,driving the development. In any case, these assumptions must be validated.

C.3.1 Evaluating the quality of HA output

The quality of the HA output depends on 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, along with the constraints on the system and its environment, which would enable identification of all hazards.
2. Identify all contributory hazards, along with the constraints on the system and its environment, which would enable identification of all contributory hazards.
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 on identifying the associated critical conditions, namely:

1. A specification of the competence required to apply the technique, so that the competence of personnel using the technique to perform HA can be evaluated with consistency.
2. A specification of the information required to apply the technique, so that the object of analysis can be evaluated with consistency.

Criteria to evaluate HA output:[116]

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 extraordinary 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" [4] is a means of tracking an item from initial expression to final disposition and closure. An entry in a hazard log is never deleted. All of 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:

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; and
   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., Identified change needed to eliminate hazard);
   2.4. Basis to allow closure; for example:
      2.4.1. Evaluation revealed that hazard control is already in place;

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

2.4.2. Evaluation resulted in recharacterization of the hazard (another entry in the hazard log); and
2.4.3. Addition of a constraint or derived requirement in the system engineering activities);
2.5. Date of closure; and
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 might (a) assist in the evaluation of some other item in the hazard log or (b) 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 level of severity of the loss of a safety function is the highest-level and,
- For systemic causes, the analysis first seeks to correctly identify hazards that would lead to the loss of a safety function and then pursues their elimination or avoidance, as explained next.

In practice, a "quick" filtering or screening evaluation (e.g., see 2.4.1 and 2.4.2 in Section C.3.2) 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 for the fit might 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 sequential order in the contributory path is found), a corresponding HA task is formulated and sequenced in accordance with its order in the contributory path.

As the evaluation of a logged item progresses, it might expose inadequacies or uncertainties in the information about the object being analyzed. Figure 10 depicts a structure for reasoning (adapted from [8]) about these uncertainties[117]. Suppose that the HA team is considering an assertion that a result of the HA (e.g., formulation of a constraint on the object being analyzed) will control the logged (contributory) hazard. Afterward, the team clarifies its reasoning through discussion, evoking challenges to the assertion and rebuttals to the challenges. The discussion might also reveal inconsistencies in the reasoning. In this manner, the team identifies factors affecting the validity of the 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}

[117] Appendix F explains how the process is applied to cross-cultural (e.g., interdisciplinary or inter-organizational) communication.

The results are recorded, showing how the assertion is supported by the evidence[118], identifying the inference rule to support the reasoning, and the technical basis for the rule (such as a causal model).[119]

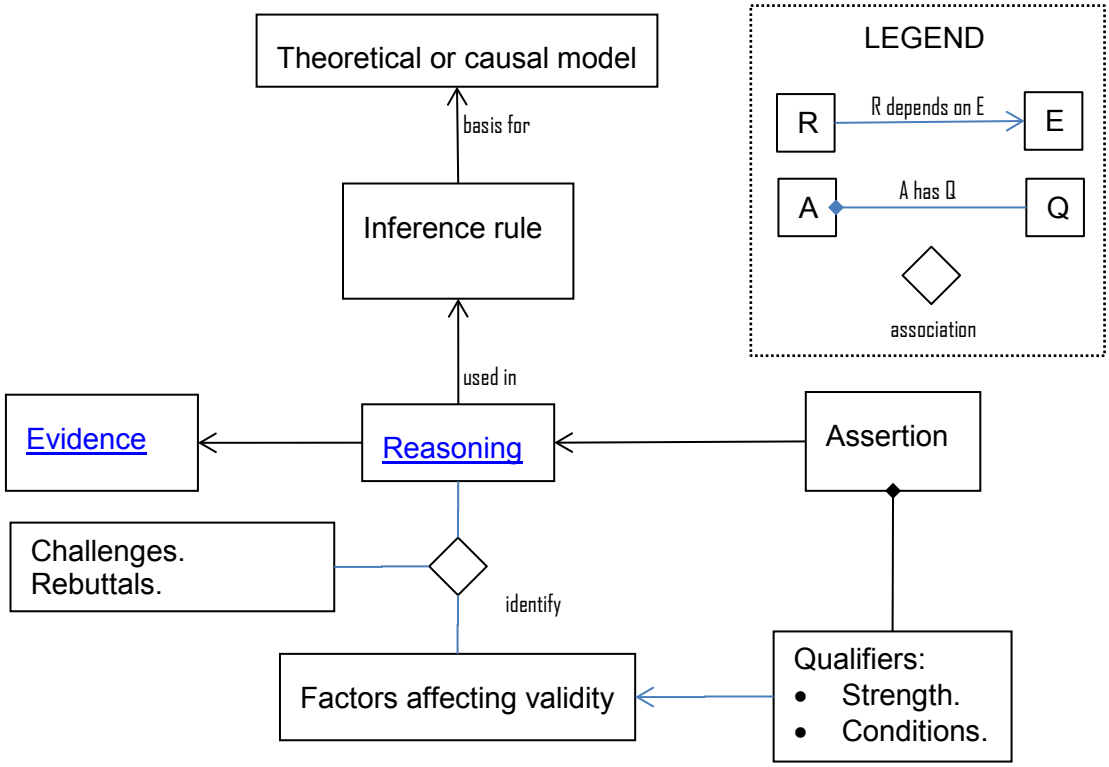

Figure 10: Structure of reasoning about the contribution to a hazard.

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, mentioned in 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 [9], in which reference *7* is a technical competence framework developed through wide consultation in the UK.

[118] This is termed “grounds” in [8].

[119] This is termed “backing” in [8].

Competence is a critical factor [10]; for example, it is recognized in the following conditions to reduce hazard spaces:

- H-0-2G1 and H-0-3G1 in Table 1
- H-culture-6G3 in Table 3
- H-SRE-1G{1, 2, and 3} in Table 10.

Competence to perform HA of an NPP digital safety system includes a complement of the following capabilities (not necessarily in one person):

1. Proven ability to learn[120], 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 the 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. Self-driven update of professional knowledge through training. Examples could include the following:
   3.1. In the application domain:
      3.1.1. How an NPP works (energy conversion from fuel to power on the grid);
      3.1.2. Heat exchange;
      3.1.3. Critical functional elements, processes, and process-state variables in an NPP;
      3.1.4. Interdependencies of items in 3.1.3;
      3.1.5. Associated (contributory) hazards; and
      3.1.6. Study of operating experience (event reports and root-cause analysis reports).
   3.2. In the industrial automation domain:
      3.2.1. Elements for sensing, actuation, and computation;
      3.2.2. Control logic;
      3.2.3. Communication;
      3.2.4. Software/firmware;
      3.2.5. Power;
      3.2.6. Associated (contributory) hazards; and
      3.2.7. Study of operating experience (event reports and root-cause analysis reports).
   3.3. Science and engineering of distributed systems, including:
      3.3.1. Computation;

[120] When the object being analyzed has some characteristic which the analyst has not encountered in past experience, as is often the case in digital safety systems, the analyst is able to acquire the needed knowledge with own initiative.

3.3.2. Communication; and
3.3.3. Storage or buffering.

3.4. Hazard and safety analysis; assurance methods and techniques for such systems.

4. Competence gained through experience in analysis of systems similar in criticality, functionality, and configuration and evidenced through:
   4.1. Good performance under the guidance of an expert in hazard analysis.
   4.2. Good performance independently.
5. Strongly safety-conscious. See Sections F.1 and F.3 of Appendix F.
6. Communication skills in group activities (see Section F.4 of Appendix F), for example:
   6.1. Ability to communicate effectively and objectively with stakeholders; 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.
   6.6. Ability to develop collective communicative competence. See Section F.4.3 of Appendix F.
7. Other interpersonal skills and characteristics that support teamwork (see Section F.4 of Appendix F). For example:
   7.1. Willingness to recognize and accept weaknesses in one's 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 (in particular, concerns or reservations).
   7.5. Ability to understand other team members' frames of reference.
   7.6. Ability to assimilate differences, neutralizing biases.
   7.7. Ability to converge[121] towards objectivity (see Figure 10). See "collective mindfulness" in Appendix F.
   7.8. Other constructive group-interaction skills.
8. Breadth and depth of competence.
   8.1. Depth: The HA team includes individuals who have mastered the engineering disciplines, technologies, products or components, processes, and dependencies involved in each phase of the system-development lifecycle. (Maintaining this depth might require different people from one development phase to the next.)
      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).

[121] Often divergent positions arise from different frames of reference, assumptions, and contexts. A competent analyst is able to recognize and articulate these underlying reasons, propose a unifying framework, and formulate qualifiers on the assertions in dispute.

8.2. Breadth:[122] Individuals are able to understand how their respective roles fit into the overall HA, including the associated interdependencies.
  8.2.1. Knowledge of the environment[123] 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 constraints to avoid or eliminate contributory hazards.
  8.2.4. Experience commensurate with the functionality and configuration of the system.

9. The HA team has cultural diversity,[124] and is able to use it to support safety.

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 the state of the practice (best-in-class implementations) and the state of the art (leading-edge implementations, not yet scaled up) in HA.

Table 22: Characterization of information richness in phase work products

Row ID	Work product of lifecycle phase	Common practice	Examples of the state of the practice (the best in the class)	Examples of the state of the art
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. [11]	Use-case scenarios [12].
			SpecTRM-RL [13]	Framework for specification & analysis [14].
			Requirements-engineering support in Naval Research Labs [15]. Requirements tables as used for Darlington NPP [16][17]. Models to support mechanized reasoning such as SysML [18].	
2	Plans (such as a safety plan, V&V plan, and HA plan)	Contain a low level of detail and are produced relatively late in the lifecycle.	V&V plan. [1] Safety plan. [19] through [21]	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 [22]. (See note 1.) SysML [18], AADL [23], and Extended EAST-ADL [24].	META [25].
4	Requirements of digital safety system	See row 1.	See row 1.	See row 1.

[122] Providing continuity to the HA team across lifecycle phases.

[123] Also see Section 2.4.1.

[124] See reference frames in item 7.5.; examples would be belief systems, values, thought processes, paradigms, customs, conventions, and language.

Row ID	Work product of lifecycle phase	Common practice	Examples of the state of the practice (the best in the class)	Examples of the state of the art
5	Architecture of digital safety system	See row 3.	See row 3.	META [25].
6	Requirements for software in digital safety system	See row 1.	[26][26][27]	See row 1.
7	Architecture for software in digital safety system	See row 3.	See row 3. MASCOT [27] and AADL [23].	META [25].
8	Detailed design of software	For application logic: Function block diagram [28]. For platform software: Combination of (a) block diagram without semantics on the symbols and (b) textual narrative.	SPARK [29][30].	META [25]. Refinement from architectural specifications.
9	Implementation of software (code)	For platform software, including communication protocols: C programming language and processor-specific assembly language	Concept of using safe subset of an implementation language: MISRA C [31][32] and language for programming FPGAs [33].	Auto-generation from detailed design.

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 on the nature of the system to be analyzed and the quality of the information contained in the various intermediate work products, as 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 “Remark(s)” column identifies concepts found useful and limitations that might confront an average practitioner. However, the adaptations of HAZOP(S) devised to evolve newer techniques require extraordinary ingenuity; utility of the adaptations depends heavily on the skills of the analysts.

When HA is applied to an early concept phase, it is called preliminary hazard analysis (PHA) [34][35].

For a broad survey of HA techniques, see [5], [36], and [37], and for additional guidance, see [38] through [42]. For a tutorial overview of HA in relation to safety-critical system development, see [43]. References [5], [36] through [42], and [43] are not included in Table 23 for a technique for which technique-specific references are listed.

Table 23: Hazard analysis techniques relevant to NPP digital safety systems

HA technique		Reference(s)	Remark(s)
Acronym	**Expanded name**		
HAZOP(S)	Hazard and Operability Studies	[44][45][46]	Concept of using teamwork, aided by HAZOP process expert. Systematizing inquiry through keywords. Systematizing understanding effects through understanding the associated deviations.
FTA	Fault-Tree Analysis	[47][48][49]	Representation and understanding of fault propagation paths as branches of a tree.
DFMEA	Design Failure Mode and Effects Analysis	[50][51][52][53]	Representation of the behavior of a failed hardware component in order to understand its effect without requiring knowledge of its internals.
FFMEA	Functional Failure Mode and Effects Analysis	[52]	Understanding the effect of the unwanted behavior of a function of the system without requiring knowledge of the system's internals. Useful in the concept phase. However, the term, failure, might limit the analyst's scope.
FuHA	Functional Hazard Analysis	[5]	It can be started in the concept phase (performed as preliminary hazard analysis) and allows for hazards arising from reasons other than failures.
FHA	Fault Hazard Analysis	[36][39][42]	Allows for hazardous conditions that arise even when no single component fails.
CCA	Cause Consequence Analysis	[36][42]	Concept of using causality model to understand fault propagation paths. Analyst may encounter difficulty when there is no single cause, but a number of contributory factors.
W/IA	What-If Analysis	[40][42]	More free-wheeling than FuHA.
CCFA	Common-Cause Failure Analysis	[36][39][42]	Requires sufficient design definition to identify common causes. The term, failure, might limit the analyst's scope. Analyst may encounter difficulty when there is no single cause, but a number of contributory factors.
HACCP	Hazard Analysis & Critical Control Points	[54]	Concept of focusing on critical process variables that affect the outcome.
SHARD	Software Hazard Analysis and Resolution	[45]	Adaptation of HAZOP to software through customization of the keywords.
FPTN/FPTC	Fault Propagation and Transformation Network/Calculus	[55]	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	[56] through [58]	Behavior modeling of the system in the finite state machine paradigm facilitates or enables: • Mathematical underpinning. • Analysis of the system's interactions with its environment. • Analysis of dynamic behavior across its elements. • Mechanized traversal. • Mechanized reasoning, especially if a directed cyclic graph is used.
STPA	System-Theoretic Process Approach	[59] through [62]	• Applicable at concept phase (without a finished design). • Applicable to understanding of systems of organizational culture.

HAZOP(S) has been adapted to analyze software [45]; this adaptation has been extended to data-flow-oriented software architecture [45], and, later, extended to systems with feedback and systems in which the initial fault is transformed into other faults as it propagates [64][64]. These

concepts and principles have influenced the AADL [23] error annex, which supports analysis of fault propagation.

Recently, a technique similar to the adaptations of HAZOP mentioned above, namely STPA, has been demonstrated in NPP applications [59][60][61][62]. For a comparative experimental study comparing STPA with five techniques, see [59][60].

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 [57], reducing the dependence on extremely high competence. However, model-based approaches introduce their own contributory hazards; the analysis for these contributors requires highly specialized competence.

See [64] for HA of device interfaces.

For an example of showing freedom from exceptions in software implementations (which are contributing hazards), in addition to showing conformance to specifications, see [30].

Static analysis tools such as [30] identify data, information, and control flow dependencies in software.

Hazard analysis can be integrated with development using a tool-supported, model-driven engineering environment such as the AADL framework [65]. Its Error Model Annex can be used to annotate the AADL model of an embedded system to support a FTA and FMEA [66][67]. Researchers are extending this environment to support STPA [68]. Also see Appendix K for direction of advancement in modeling dependencies for hazard analysis.

C.7 References for Appendix C

[1] Institute of Electrical and Electronics Engineers (IEEE), IEEE Standard 1012-2012, “IEEE Standard for System and Software Verification and Validation,” Piscataway, NJ, March 29, 2012.

[2] U.S. Department of Defense (DoD), Joint Software System Safety Committee, “Software System Safety Handbook: A Technical & Management Team Approach,” Washington, DC, December 1999. Available at http://www.system-safety.org/Documents/Software_System_Safety_Handbook.pdf.

[3] DoD, “Department of Defense Standard Practice: System Safety,” MIL-STD-882E, Washington, DC, May 11, 2012, available at http://www.system-safety.org/Documents/MIL-STD-882E.pdf.

[4] UK Ministry of Defence, “Hazard Log,” Procedure SMP 11, v. 2.2s, London, UK, November 2007, available at https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/27584/SMP11v22final.pdf.

[5] Society of Automotive Engineers (SAE), “Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment,” ARP4761, Warrendale, PA, December 1, 1996.

[6] McDermid, J.A., “Safety Critical Software,” Chapter 506 of Blockley, R., and W. Shyy, eds., *Encyclopedia of Aerospace Engineering*, Hoboken, NJ: John Wiley & Sons, December 2010 (added to the encyclopedia as an online publication on June 15, 2012, accessible through DOI:10.1002/9780470686652.eae506).

[7] Hawkins, R.D., I. Habli, and T.P. Kelly, "The Principles of Software Safety Assurance," *Proceedings of the 31st International System Safety Conference, Boston, MA*, *August 12–16, 2013,* International System Safety Society, Unionville, VA, 2013.

[8] Toulmin, S, *The Uses of Argument*, Cambridge, UK: Cambridge University Press, 1958.

[9] UK Health and Safety Executive (HSE), "Managing Competence for Safety-related Systems," available at http://www.hse.gov.uk/consult/condocs/competence.pdf.

[10] Chambers, L., "A Hazard Analysis of Human Factors in Safety-Critical Systems Engineering," *Proceedings of the 10th Australian Workshop on Safety-Critical Systems and Software, Vol. 55, 2006, Sydney, Australia*, Australian Computer Society, Darlinghurst, Australia, 2005.

[11] Hinchey, A.G., J.L. Rash, and C.A. Rouff, "Towards an automated development methodology for dependable systems with application to sensor networks," *Proceedings of the 24th International Performance, Computing, and Communications Conference, Phoenix, AZ, April 7–9, 2005*, IEEE, Piscataway, NJ, 2005, pp. 445–451.

[12] Allenby, K., and T. Kelly, "Deriving Safety Requirements Using Scenarios," *Proceedings of the Fifth International Symposium on Requirements Engineering (ISCE), Toronto, ON, August 27–31, 2001*, IEEE, Piscataway, NJ, 2001, August 7, 2002, pp. 228–235.

[13] Safeware Engineering Corporation, "Safeware Engineering Corporation: SpecTRM Features," available at http://www.safeware-eng.com/software safety products/features.htm.

[14] Day, N.A., and J.A. Joyce, "A framework for multi-notation requirements specification and analysis," *Proceedings of the Fourth International Conference on Requirements Engineering, Schaumburg, IL, June 19–23, 2000*, IEEE, Piscataway, NJ, 2000, available at http://ieeexplore.ieee.org/ielx5/6907/18574/00855551.pdf?tp=&arnumber=855551&isnumber=18574.

[15] Heitmeyer, C., J. Kirby, and B. Labaw, "The SCR method for Formally Specifying, Verifying, and Validating Requirements: Tool Support," *Proceedings of the 19th International Conference on Software Engineering, Boston, MA, May 17–23, 1997,* Association for Computing Machinery (ACM) and IEEE, New York, NY, and Piscataway, NJ, 1997, pp. 610–611, available at http://ieeexplore.ieee.org/ielx3/4837/13372/00610430.pdf?tp=&arnumber=610430&isnumber=13372.

[16] Parnas, D.L., and J. Madey, "Functional Documents for Computer Systems," *Science of Computer Programming* 25(1):41–61, October 1995.

[17] Galloway, A., et al., "On the Formal Development of Safety-Critical Software," in Meyer, B., and J.C.P. Woodcock, eds., *Verified Software: Theories, Tools, Experiments, First IFIP TC 2/WG 2.3 Conference, VSTTE 2005, Zurich, Switzerland, October 10–13, 2005*, pp. 362–373.

[18] SysML.org, "SysML.org: SysML Open Source Specification Project," available at http://www.sysml.org/.

[19] International Organization for Standardization (ISO), "Road Vehicles—Functional safety—Part 2: Management of functional safety," ISO 26262-2:2011, Geneva, Switzerland, 2011.

[20] ISO, "Road Vehicles—Functional safety—Part 3: Concept phase," ISO 26262-3:2011, Geneva, Switzerland, 2011.

[21] ISO, "Road Vehicles—Functional safety—Part 4: Product development at the system level," ISO 26262-4:2011, Geneva, Switzerland, 2011.

[22] Despotou, G., R. Alexander, and T.P. Kelly, "Addressing Challenges of Hazard Analysis in Systems of Systems," *Proceedings of the 3rd Annual IEEE International Systems Conference, Vancouver, BC, March 23–26, 2009*, IEEE, Piscataway, NJ, 2009.

[23] Carnegie-Mellon University, "AADL predictable model-based engineering," available at http://www.aadl.info/aadl/currentsite/.

[24] Mader, R., et al., "A Computer-Aided Approach to Preliminary Hazard Analysis for Automotive Embedded Systems," *Proceedings of the 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, Las Vegas, NV, April 27–29, 2011*, IEEE, Piscataway, NJ, 2011, pp. 169-178.

[25] Kable Intelligence Limited, "Vanderbilt University to support META tools maturation for DARPA AVM programme - Army Technology," available at http://www.army-technology.com/news/newsvanderbilt-university-support-meta-tools-maturation-darpa-avm-programme.

[26] Miller, S.P., and A.C. Tribble, "Extending the Four-Variable Model to Bridge the System-Software Gap," *Proceedings of the 20th Digital Avionics System Conference, Daytona Beach, FL, October 14–18, 2001*, Volume 1, IEEE, Piscataway, NJ, pp. 4E5/1–4E5/11.

[27] Simpson, H.R., "The Mascot method," *Software Engineering Journal* 1(3):103–120, May 1986.

[28] IEC, "Programmable controllers—Part 3: Programming languages" IEC 61131-3, Edition 3.0, Geneva, Switzerland, 2013.

[29] Barnes, J.G.P., *High Integrity Software: The SPARK Approach to Safety and Security*, Boston: Addison-Wesley, April 2003.

[30] AdaCore, "AdaCore: SPARK Pro," available at https://www.adacore.com/sparkpro/.

[31] MISRA Ltd., "MISRA C: 2012," available at http://www.misra.org.uk/MISRAHome/MISRAC2012/tabid/196/Default.aspx.

[32] LDRA, MISRA C / MISRA C++ Coding Standards Compliance see: http://www.ldra.com/en/software-quality-test-tools/group/by-coding-standard/misra-c-c.

[33] Conmy, P.M., C. Pygott, and I. Bate, "VHDL Guidance for Safe and Certifiable FPGA Design," 5th International Conference on System Safety, Manchester, UK, October 18–20, 2010, The Institution of Engineering and Technology, Stevenage, UK, 2010, pp. 1–6.

[34] Gowen, L.D., J.S. Collofello, and F.W. Calliss, "Preliminary Hazard Analysis for Safety-Critical Software Systems," *Proceedings of the Eleventh Annual International Phoenix Conference on Computers and Communications, Scottsdale, AZ, April 1–3, 1992*, IEEE, Piscataway, NJ, 1992, pp. 501–508.

[35] Safeware Engineering Corporation, "Safeware Engineering Corporation: White Papers - Preliminary Hazard Analysis," available at http://www.safeware-eng.com/Safety White Papers/Preliminary Hazard Analysis.htm.

[36] Ericson II, C.A., *Hazard Analysis Techniques for System Safety*, Hoboken, NJ: John Wiley & Sons, August 2005.

[37] NRC, "Software Safety Hazard Analysis," NUREG/CR-6430, February 1996, ADAMS Public Legacy Library Accession No. 9602290270.

[38] National Aeronautics and Space Administration (NASA), "NASA Software Safety Guidebook", NASA-GB-8719.13, Washington, DC, March 31, 2004, available at http://www.hq.nasa.gov/office/codeq/doctree/871913.pdf.

[39] U.S. Air Force, "Air Force System Safety Handbook", Kirtland AFB, NM, July 2000, available at http://www.system-safety.org/Documents/AF_System-Safety-HNDBK.pdf.

[40] Maragakis, I., et al., "Guidance on Hazards Identification," European Commercial Aviation Safety Team, Cologne, Germany, March 2009, available at http://www.easa.europa.eu/essi/documents/ECASTSMSWG-GuidanceonHazardIdentification.pdf.

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

[42] "System Safety Analysis Handbook," Second Edition, International System Safety Society, Unionville, VA, 1999.

[43] Johnson, C.W., "Safety Critical Systems Development: Part II of the Notes," University of Glasgow, Glasgow, UK, October 2006, available at http://www.dcs.gla.ac.uk/~johnson/teaching/safety/slides/pt2.pdf.

[44] Chemical Industries Association, Chemical Industry Safety and Health Council (CISHEC), "A Guide to Hazard and Operability Studies," London, UK, 1977.

[45] Redmill, F., M. Chudleigh, and J. Catmur, *System Safety: HAZOP and Software HAZOP*, Hoboken, NJ: John Wiley & Sons, June 1999.

[46] International Electrotechnical Commission (IEC), "Hazard and Operability Studies (HAZOP Studies)—Application Guide," IEC 61882:2001, Edition 1.0, Geneva, Switzerland, 2001.

[47] Vesely, W.E., et al, "Fault Tree Handbook," NUREG-0492, January 1981, Agencywide Documents Access and Management System (ADAMS) Accession No. ML12167A103.

[48] NASA, "Fault Tree Handbook with Aerospace Applications," Version 1.1, Washington, DC, August 2002, available at http://www.hq.nasa.gov/office/codeq/doctree/fthb.pdf.

[49] Park, G.-Y., et al., "Fault Tree Analysis of KNICS RPS Software," *Nuclear Engineering Technology* 40(5):397–408, August 2008.

[50] SAE, "Potential Failure Mode and Effects Analysis in Design (Design FMEA), Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes (Process FMEA), 2009" SAE J1739, Warrendale, PA, January 15, 2009.

[51] 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," Washington, DC, November 2011, available at http://rsdo.gsfc.nasa.gov/documents/Rapid-III-Documents/MAR-Reference/GSFC-FAP-322-208-FMEA-Draft.pdf.

[52] Goddard, P.L., "Software FMEA Techniques," *Proceedings of the Annual Reliability and Maintainability Symposium, Los Angeles, CA, January 24–27, 2000*, IEEE, Piscataway, NJ, 2000, pp. 118–123.

[53] Park, G.-Y. "Software FMEA Analysis for Safety Software," *Proceedings of the 17th International Conference on Nuclear Engineering, Brussels, Belgium, July 12–16, 2009*, Volume 5, ASME, New York, NY, 2009, pp. 831–837.

[54] U.S. Food and Drug Administration, "Hazard analysis & critical control points (HACCP) > HACCP Principles & Application Guidelines," available at http://www.fda.gov/Food/GuidanceRegulation/HACCP/ucm2006801.htm.

[55] Fenelon P., et al., "Towards Integrated Safety Analysis and Design," *ACM Applied Computing Review* 2(1):21–32, March 1994.

[56] Garrett, C., and G. Apostolakis, "Context in the risk assessment of digital systems," *Risk Analysis* 19(1):23–32, February 1999.

[57] Garrett, C. and G. Apostolakis, "Automated hazard analysis of digital control systems," *Reliability Engineering and System Safety* 77(1):1-17, July 2002.

[58] NRC, "A Benchmark Implementation of Two Dynamic Methodologies for the Reliability Modeling of Digital Instrumentation and Control Systems," NUREG/CR-6985, February 2009, ADAMS Accession No. ML090750687.

[59] Torok, R., and B. Geddes, "Systems Theoretic Process Analysis (STPA) Applied to a Nuclear Power Plant Control System," Electric Power Research Institute, Palo Alto, CA, 2013, available at http://psas.scripts.mit.edu/home/wp-content/uploads/2013/04/02_EPRI_MIT_STAMP_Mar2013.pdf.

[60] Torok, R., "Hazard Analysis Methods for Digital Instrumentation and Control Systems," 3002000509, Electric Power Research Institute, Palo Alto, CA,June 2013, publicly available for fee at http://www.epri.com (non-publicly available at ADAMS Accession No. ML13234A512)

[61] Song, Y., "Applying system-theoretic accident model and processes (STAMP) to hazard analysis," Master's thesis, McMaster University, Hamilton, ON, available at http://hdl.handle.net/11375/11867.

[62] Thomas, J., F.L. de Lemos, and N. Leveson, "Evaluating the Safety of Digital Instrumentation and Control Systems in Nuclear Power Plants," NRC-HQ-11-6-04-0060, Massachusetts Institute of Technology, Cambridge, MA, November 2012, available at http://sunnyday.mit.edu/papers/MIT-Research-Report-NRC-7-28.pdf.

[63] McDermid, J.A., et al., "Experience with the application of HAZOP to computer-based systems," *COMPASS '95: Proceedings of the Tenth Annual Conference on Computer Assurance, June 25–29, 1995, Gaithersburg, MD*, IEEE, Piscataway, NJ, pp. 37–48.

[64] Wallace, M., "Modular Architectural Representation and Analysis of Fault Propagation and Transformation," *Electronic Notes in Theoretical Computer Science* 141(3):53–71, December 1, 2005.

[65] Feiler, P.H., D.P. Cluch, and J.J. Hudak, "The Architecture Analysis & Design Language (AADL): An Introduction," Technical Note CMU/SEI-2006-TN-011, Software Engineering Institute, Pittsburgh, PA, February 2006, available at http://www.sei.cmu.edu/reports/06tn011.pdf.

[66] Feiler, P., and A. Rugina, "Dependability Modeling with the Architecture Analysis & Design Language (AADL)," CMU/SEI-2007-TN-043, Software Engineering Institute, Pittsburgh, PA, July 2007, available at http://www.sei.cmu.edu/reports/07tn043.pdf.

[67] Delange, J., and P. Feiler, “Supporting Safety Evaluation Process using AADL,” *Proceedings of the 7th Layered Assurance Workshop, New Orleans, LA, December 9–10, 2013*, Applied Computer Security Associates, Silver Spring, MD, available at http://www.acsac.org/2013/workshops/law/2013-law-proceedings.pdf.

[68] Sam Procter and John Hatcliff, "An Architecturally-Integrated, Systems-Based Hazard Analysis for Medical Applications", 12th ACM/IEEE International Conference on Formal Methods and Models for Codesign, (MEMOCODE 2014), Lausanne, Switzerland, October 19-21, 2014.

APPENDIX D: Refinement

Enabling verifiability earlier in the lifecycle through stepwise refinement

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

Integrative editing by Sushil Birla

D.1 Purpose and Scope

This appendix explains refinement (see Section D.2) as an enabler for the verifiability and, thus, the assurability of a system (see item H-S-1.1G1.4 in Table 8).

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 in which a specification is expressed through an informal language and informal diagrams. Such a specification might be ambiguous and its meaning might differ, depending on individual subjective judgment, as illustrated in the following situation:

> When a system[125] 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[126], correct set of requirements specification may require engineering activities (e.g., elicitation; system-level hazard analysis; validation) other than refinement [1].

This rigorous form of refinement reduces sources of uncertainty in the verification process. This benefit is further discussed in Section D.3 and the corresponding required restrictions are introduced in Section D.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, in which each successive step refines its input specification ([4] and [5]). Each transformation step entails some design decisions [6]. In other words, it reduces the design space for the subsequent steps.

---

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

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

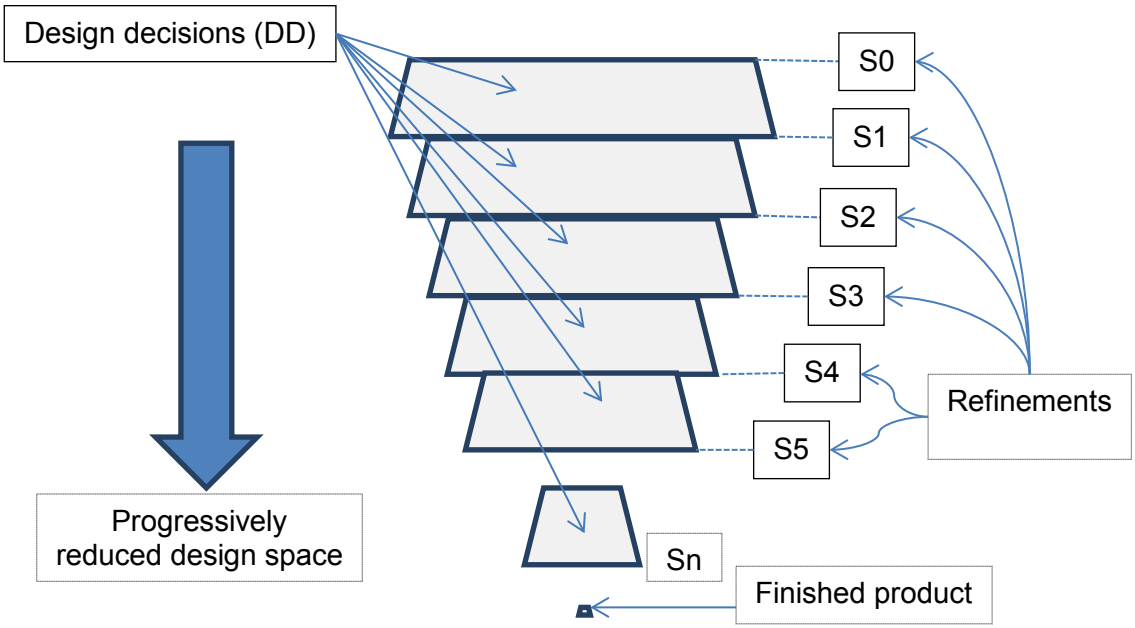

Figure 11: Stepwise refinement: design decisions are made in small steps.

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 data or 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, to which each element can be sent (written) or from which each element can be retrieved (read) as a value of length in SI units, but in which each element is stored in a compact form.

Type of data or information	Example of abstract level	Example of refined level
Structured data	Location of 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 and location is designated A_{C0}.
Function	Calculate the location of A with respect to another Cartesian coordinate frame C1 using IEEE 754, "IEEE Standard for Floating-Point Arithmetic"; the result is designated A_{C1}.	Calculate location of A_{C1} using matrix representation[127] and matrix functions that conform to IEEE 754: $[A_{C1}] = [A_{C0}] - [C1_{C0}]$.

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 [1]. Refinement (in the rigorous sense as mentioned in Section D.4) enables "verification by construction" that the original specification and initial constraints are satisfied [3].

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, which are not needed but might render the analysis infeasible. The abstraction has to suit the analytical purpose.

The enabling idea in the transformation from the abstract to the refined specification is that the verification performed on the abstract level remains valid for the refined specification. This idea can be applied to a sequence of refinement steps: Verification of properties successfully applied to abstractions also holds 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 for making a sequence of design decisions that rules out unsafe choices or choices for which safety cannot be assured (e.g., because the technological basis does not exist or 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 it is used with mathematical underpinnings such as 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 in such a way that the specification for the "chunk" is preserved [8].

[127] In this case, the square brackets [] represent a matrix.

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 correctness by construction in the refinement process.

D.4.1 Refinement as logical implication

Logically, refinement corresponds to *implication*; the refined specification satisfies the 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 "$\Leftarrow$" 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

The *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, expressed as

$S0 \Leftarrow S1$,

and S2 is a refinement of S1, expressed as

$S1 \Leftarrow S2$,

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 ,ust be comprehensible. 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 $\mathrm{M}_1, \mathrm{M}_2, \ldots, \mathrm{M}_k$.

D.4.4.1 Composing and Decomposing Interfaces

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

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

The operator $\otimes$ induces a composition operation on specifications [9].

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

$$\mathrm{M} = \mathrm{M}_1 \otimes \mathrm{M}_2 \otimes \ldots \otimes \mathrm{M}_k$$

This equation expresses both that M is the result of composing the elements $\mathrm{M}_1, \mathrm{M}_2, \ldots \mathrm{M}_k$ and that M may be correctly decomposed into the elements $\mathrm{M}_1, \mathrm{M}_2, \ldots \mathrm{M}_k$.

Following this scheme, a specification S is decomposed into a number of specifications $S_1, S_2, \ldots S_k$ 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 represented by $\mathrm{M}_1 \otimes \mathrm{M}_2 \otimes \ldots \otimes \mathrm{M}_k$ where the models of the elements $\mathrm{M}_1, \mathrm{M}_2, \ldots \mathrm{M}_k$ fulfill the specifications $\mathrm{S}_1, \mathrm{S}_2, \ldots \mathrm{S}_k$ respectively.

Such a step of decomposition of a system specification into specifications 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][10][11][12]. 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 also the refinement

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

and given refinements $\mathcal{R}_1$, $\mathcal{R}_2$ … $\mathcal{R}_k$ of the specification S_1, S_2 … S_k:

If the refinement relation is compositional for composition S, we can conclude that:

$$S_1 \otimes S_2 \otimes \ldots \otimes S_k \Leftarrow \mathcal{R}_1 \otimes \mathcal{R}_2 \otimes \ldots \otimes \mathcal{R}_k$$

and, by transitivity of refinement,

$$S \Leftarrow \mathcal{R}_1 \otimes \mathcal{R}_2 \otimes \ldots \otimes \mathcal{R}_k$$

Compositional refinement also captures the idea of **compatibility** (replaceability) of a system or its elements. Consider a system given by a composition of elements, such that the system design is correct as long as the elements satisfy their respective 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 by its model M; its behavior is represented by its specification S. The system model is decomposed into modeling elements $\mathrm{M1}$, $\mathrm{M2}$, and $\mathrm{M3}$ and their respective behaviors are represented by S_1, S_2, and S_3. If 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 $\mathrm{M1}$, $\mathrm{M2}$, and $\mathrm{M3}$ specified by S_1, S_2, and S_3.

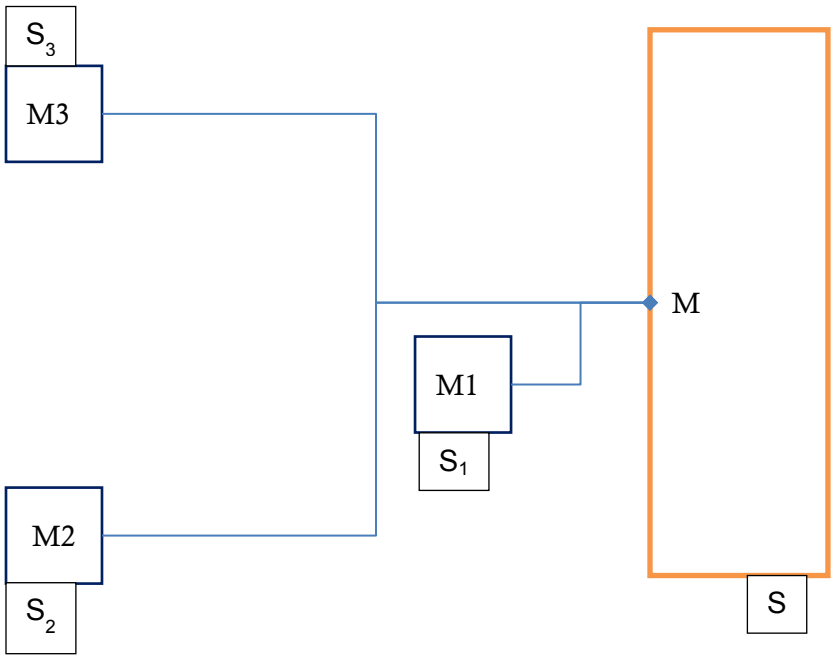

Figure 12: Example of architectural refinement through decomposition.

D.5 References for Appendix D

[1] Jackson, M., "Refinement, Problems and Structures" (extended abstract) in *Proceedings of Dagstuhl Seminar 09381: Refinement Based Methods for the Construction of Dependable Systems, 13–18 September 2009*, Saarbrücken, Germany: Schloss Dagstuhl – LZI, 2009, available at http://mcs.open.ac.uk/mj665/Dagstuhl09ExtAbst.pdf.

[2] ISO, IEC, and IEEE, "Systems and software engineering—Vocabulary," ISO/IEC/IEEE 24765:2010, Piscataway, NJ, 2010.

[3] Woodcock, J., and J. Davies, *Using Z: Specification, Refinement, and Proof*, Upper Saddle River, NJ: Prentice Hall, May 1996.

[4] Butler, M.J., "Stepwise refinement of communicating systems," *Science of Computer Programming* 27(2):139–173, September 1996.

[5] de Bakker, J.W., W.-P. de Roever, and G. Rozenberg, eds., *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness, Proceedings of the REX Workshop, Mook, The Netherlands, May/June 1989*, Berlin: Springer, 1990.

[6] Wirth, N., Program development by stepwise refinement," *Communications of the ACM* 14(4):221–227, April 1971.

[7] Morris, J.M., "Laws of data refinement," *Acta Informatica* 26(4):287–308, February 1989.

[8] Morgan, C., *Programming from Specifications*, 2nd Edition, Upper Saddle River, NJ: Prentice Hall, 1994.

[9] Broy, M., "A Theory for Requirements Specification and Architecture Design of Multi-Functional Software Systems," in Liu, Z., and H. Jifeng, eds., *Mathematical Frameworks for Component Software: Models for Analysis and Synthesis*, Singapore: World Scientific Publishing, 2006, pp. 119–154.

[10] Broy, M., “Compositional refinement of interactive systems,” *Journal of the ACM* 44(6):850–891, November 1997

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

[12] Broy, M., “A Logical Basis for Component-Oriented Software and Systems Engineering,” *The Computer Journal* 53(10):1758–1782, February 2010.

APPENDIX E: Checklists to assist hazard recognition

This appendix is a collection of checklists assimilated from a variety of sources such as [1], [2], and [3]. It is not an exhaustive coverage of hazard sources, categories, or groupings relevant to a nuclear power plant's (NPP's) 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 Appendix D to National Aeronautics and Space Administration (NASA) Reference Publication 1358 [1] and is organized by categories of hazard origination or source. For each category, Table 25 identifies a variety of effects which might lead to loss.

Table 25: Some categories of hazard origination

Category of hazard origination	Effect which might 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
	Wearing out
	Maintenance error
	Result of activity of organisms:
	• Animals, such as:
	  ○ Vermin
	  ○ Varmints
	  ○ Mud daubers
	• Trees
	  ○ Invasion of roots
	  ○ Congestion from leaves.

Category of hazard origination	Effect which might lead to loss
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 Lighting strike Grounding failure Inadvertent activation
Electrical	Shock Burns Overheating Ignition of combustibles Inadvertent activation Power outage Distribution backfeed 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	Overpressurization 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)

Category of hazard origination	Effect which might lead to loss
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 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

Category of hazard origination	Effect which might lead to loss
Physiological (see “Ergonomic”)	Temperature extremes Nuisance dusts/odors Barometric pressure extremes Fatigue Lifted weights Noise Vibration (Raynaud’s syndrome) Mutagens Asphyxiants Allergens Pathogens Radiation (see “Radiation”) Cryogens Carcinogens Teratogens Toxins Irritants
Human Factors (see “Ergonomic”)	Operator error Inadvertent operation Failure to operate Operation too early or late Operation out of sequence Right operation but wrong control Operated too long Operated too briefly
Ergonomic (see “Human Factors”)	Fatigue Inaccessibility Nonexistent/inadequate “kill” switches Glare Inadequate control/readout differentiation Inappropriate control/readout labeling Faulty workstation design Inadequate/improper illumination
Utility Outages: • Unannounced; unscheduled • Sudden • Unexpected; unforeseen	Electricity Steam Heating/cooling Ventilation Air conditioning Compressed air/gas Lubricant drains/sumps Fuel Exhaust
Mission Phasing	Transport Delivery Installation Calibration Checkout Shakedown 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 [2] 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 for hazards, assimilated from a variety of sources such as [2]:

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, and fans
19. Rotating machinery
20. Actuating devices
21. Nuclear

E.3 Checklist of hazard sources in semiconductor manufacturing

Table 26 is a set of examples from the semiconductor manufacturing industry [3], 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 might 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

Categories of hazard sources	Potential loss or effect which might lead to loss
Human Hazards Hazards to perception (inadequate control/display identification), dexterity (inaccessible control location), life support, and error probability (inadequate data for decisionmaking). Hazardous conditions caused by position (hazardous location/height), equipment (inadequate visual/audible warnings or heavy lifting), or other elements that could cause injury to personnel.	Personnel injury: Skin abrasion, cuts, bruises, burns, falls, etc. Muscle/bone damage Sensory degradation or loss Death Equipment damage by improper operation/handling might also occur
Kinetic/Mechanical Energy (Acceleration) System/component linear or rotary motion. Change in velocity or impact energy of vehicles, components, or fluids.	Impact Disintegration of rotating components Displacement of parts or piping Seating or unseating of valves or electrical contact Detonation of shock-sensitive explosives Disruption of metering equipment Friction between moving surfaces
Material Deformation Degradation of material because of an external catalyst (e.g., corrosion, aging, embrittlement, fatigue, etc.).	Change in physical or chemical properties: corrosion, aging, embrittlement, oxidation, etc. Structural failure Delamination of layered material Breakdown of electrical insulation
Natural Environment Conditions including lighting, wind, flood, temperature extremes, pressure, gravity, humidity, etc.	Structural damage from wind Equipment damage Personnel injury
Pressure Potential energy of a system or component (e.g., a fluid system or air system), including high, low, or changing pressure.	Blast/fragmentation from container overpressure rupture Line/hose whipping Container implosion System leaks Aero-embolism, bends, choking, or shock Uncontrolled pressure changes in air/fluid systems
Radiation Conditions involving electromagnetic, ionizing, thermal, or ultraviolet radiation (including that from lasers and optical fibers).	Uncontrolled initiation of safety control systems and 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	Damage to, irritation of, or other effects on: Respiratory system Circulatory system Organs of the body Skin Nervous system
Vibration/Sound System/component-produced energy	Material failure Pressure/shock-wave effects Loosening of parts Chattering of valves or contacts Interference with verbal communications Degradation or failure of displays

E.4 Hazard sources in the physical environment of a digital safety system

Disruption in or emissions from the environment or physical conditions in the environment might degrade a safety function of the analyzed digital instrumentation and control (DI&C) system in an NPP in any of these ways:

1. Water in unwanted space
2. Transfer of unwanted energy in various forms; for example:
   - 2.1. Fire
   - 2.2. Lightning
   - 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; for example:
     - 2.14.1. Pulse
     - 2.14.2. Sunspot or solar flare
3. Interruption of services (primary, secondary, or other forms of backup); 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 [4]
   - 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 or 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 and 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 might affect safety adversely in any of these ways:

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 or commands); for example:
   3.1. Byzantine behavior
   3.2. Behaving like a "babbling idiot" in a connected network
4. Excessive[128] load or demand on resources; for example:
   4.1. Electric power overload caused by a short circuit
   4.2. Communication bus overload
   4.3. Locking up resources to the exclusion of other "users" of those resources.

Note: Items 3, 4.2, and 4.3 are "logical" rather than physical contributory causes.

E.6 References for Appendix E

[1] Goldberg, B.E., et al., "System Engineering 'Toolbox' for Design-Oriented Engineers," NASA Reference Publication 1358, National Aeronautics and Space Administration, Marshall Space Flight Center, AL, December 1994, available at http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19950012517.pdf.

[2] Ericson II, C.A., Hazard Analysis Techniques for System Safety, Hoboken, NJ: John Wiley & Sons, August 2005.

[3] International SEMATECH, Inc., "Hazards Analysis Guide: A Reference Manual for Analyzing Safety Hazards on Semiconductor Manufacturing Equipment," Technology Transfer # 99113846A-ENG, Austin, TX, November 30, 1999, available at http://www.sematech.org/docubase/document/3846aeng.pdf.

[4] NRO Office of Inspection & Enforcement, PNO-77-146_8-19-77, 146th Preliminary Notice of Event or Occurance for 1977, dated August 19, 1977, Legacy (microfiche) ADAMS Accession Number 9809170110, Microform Address A5119:285-A5119:285.

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

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

This appendix draws on 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 an NPP. In the absence of context-specific research, this appendix assimilates[129] information from broader fields of applicable research in support of the premise that collective mindfulness (Section F.5) within the organization is an essential factor for (a) reducing the hazard space in engineering a digital safety system for an NPP and (b) conducting the associated hazard analysis. Most of the scholarship is concerned with operations of technologically and organizationally intricate systems (such as those in nuclear power plants, aircraft carriers, aviation, the petroleum industry, occupational safety, and healthcare) ([1], [2], [3] through [5], [6], and [7]). The literature refers to them as "high-reliability organizations" (HROs), where an HRO is defined as an organization that operates (works with) hazardous (hazard-contributing) technologies without having this operation lead to a catastrophe (loss of safety).

Swanson et al. [8], 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:

Just as, in the operational environment, a “high-reliability[130] organization” operates hazardous technologies without leading to catastrophe [9], in the engineering environment, a high-quality engineering organization (HQEO) develops and maintains technological systems without entailing associated hazards.

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 using best communication practices and sound principles from the social sciences.

HQEOs, just like 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” ([9][10]).[131] These organizations resist patterns of habitual[132] thinking and communicating that might 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

[129] Assimilation includes mapping certain terms to the context of RIL-1101.

[130] “Reliability” in this context does not have the same definition as used in fault-tolerant engineered systems. To avoid the confusion, Appendix F uses the term “high-quality engineering organization.”

[131] “Hazard identification and evaluation” is implicit in the expression, “risk detection, assessment, and management” in the cited references.

[132] Interpreting new information through an old reference frame—the traditional belief system.

intricacies of the system and to help the organization avoid a hazardous condition[133] and prevent the consequential loss (e.g., degradation of a safety function). Organizational culture is a 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 organizing principles for high-quality[134] processes ([6], [9], [10], and [11]):

1. Preoccupation with hazard[135] identification: Treat every piece of information as a potential symptom that something could be wrong with the system.
2. Reluctance to accept simplistic[136] explanations and models: Always hold current mental models in question with a persistent goal to create more complete and nuanced[137] explanations and models of the system.
3. Sensitivity to operations, including situational awareness of the (current state of the) system: Be able to notice anomalies,[138] track them, and resolve them.
4. Commitment to resilience: Learn from mistakes, correcting one's perceptions to represent reality[139] well enough to identify (contributory) hazards, in order to detect, contain, and recover from mistakes.[140]
   4.1. Be able to respond to unanticipated conditions (outside the boundary of the organization's deterministic processes) without compromising its safety goal.[141]
   4.2. Be able to learn and grow from previous episodes of resilient action.
5. Deference to expertise: Cultivate diversity; delegate to (that is, empower) people who are closer to the situation and can recognize more subtle contributors to a hazard in intricate environments, and assimilate information from such people's diverse perspectives.

An HQEO practices these principles in its everyday activities. However, there are ways to measure an organization's ability to follow these principles with surveys. A survey measure of the five principles is in [9] and [10].

---

[133] The reference uses the terms "error" and "failure" for this.

[134] The references [6][9][10] use the term, "reliability"

[135] The references [6][9][10] use the term "failure."

[136] In the RIL-1101 context, this means "not adequately representative of reality; missing (contributory) hazards."

[137] In the RIL-1101 context, this means "reflecting subtle details that enable (contributory) hazard identification."

[138] In the RIL-1101 context, this maps into "(contributory) hazards."

[139] The references [6][9][10] use the term "complicating" to imply a mental model that reflects reality more accurately.

[140] The references [6][9][10] use 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.

[141] The references [6][9][10] use the expression "absorb strain and preserve functioning despite presence of adversity."

F.2 Accountability, Standardization, and Adaptation

Many assumptions associated with safety management are based in traditional "scientific" management, inheriting the following characteristics:

1. A standardization[142] of work process, output, skills, and organizational norms (e.g., safety culture).
2. A strict separation of planning[143] and operations processes.
3. The use of "scientific" measurement to develop the standards and to detect flaws in the system [24].

"Scientific" management is based on the premise that standardized processes in normal operations control output and prevent mistakes [24][12]. This premise is related to master premises that efficiency and predictability are desirable performance characteristics[144] of organizational processes [12][13]. 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") [24][13]. Thos assumption does not make adequate provision[145] for unanticipated conditions and limits the organization's ability to recognize (contributory) hazards [24][9][10][13][14].

Likewise, the desire to establish a clear hierarchical "command and control" tree derives from this assumption [12][15]. Decades of research about organizations, including the nuclear industry, clearly document that the very nature of bureaucracy in organizations diffuses accountability [12]. Hierarchical "command and control" is viewed as a strength, because bureaucracy's prescriptive "deterministic" processes enable accomplishment of organizational tasks and goals without full dependence on an individual thinking for adjusting to situation-specific unanticipated conditions [12]. Thus, individuals often base decisions on assumptions underlying the "deterministic" processes that are not always made explicit [12]. However, as noted in H-culture-9 in Table 3, an overly rigid "command and control" organization structure can increase the hazard space because the implicit assumptions and premises might not hold. The top-down allocation of roles, responsibilities and performance metrics is based on a deterministic process model, and does not make adequate provision for bottom-up observation and feedback of real conditions and adaptation to them.

Organizational research asserts that the nature of bureaucracy creates a powerful force to diffuse accountability throughout the organization [12][15]. In terms of ethics, some researchers lament this organizational force and call for organizations, in general, to become mindful of this tendency and counteract it whenever possible [12]. But rather than tracing all decisions through individual accountability, they suggest that organizational members question assumptions and premises that pervade the organizational culture [12]. The Toulmin model [16] introduced in Section C.3.3 of Appendix C 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

[142] Which includes top-down decomposition and allocation of responsibilities along the organizational (command and control) structure, down to the individual.

[143] Rigid hierarchical (top-down) plans limit local autonomy during execution or operation.

[144] The references use the term "outcomes."

[145] For example, by establishing an organizational architecture for collective mindfulness.

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 to hyperlink the rationale to supporting information and documents ([17] through [19]).

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 [20] is to raise awareness of the faulty assumption that accidents result from a lapse of "scientific" management to control employees—often referred to as "human error" [20][22]. In the context of complex, high-risk technologies, it is worth considering his argument that the nature of complex technical systems makes it extraordinarily difficult for standardization of organizational procedures to anticipate all possible combinations of mistakes. An HQEO takes 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 [3][9][10]. Requisite variety is the variation in frames of reference and knowledge that makes the organization capable of recognizing and addressing hazards [6]. 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 [14][21] [22] [23]. 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[146] and adapt the procedures ([9], [10], and [12]). 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 to the system [12][23] [24][25].

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 organizational culture and decisional premises (Section F.3), the role of communication in developing collective mindfulness and following Toulmin's model of argumentation [16] (Section F.4), and the relationship between professional identification and collective mindfulness and competence (Section F.5). Additionally, each section cites tools and techniques for 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 [23][24][25][26][27][28]. Organizational culture is a complex concept, and because of its complexity, it is difficult to define conceptually and difficult to measure [13][26][28][30]. 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" [25].

In the nuclear industry (and others), this concept is often called "safety culture," defined by the International Atomic Energy Agency (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" [24]. Thus, one important

[146] Delegate and distribute control; provide the autonomy (empower) to adapt, learn, and give feedback.

way to think about the role of organizational culture in the process of hazard analysis is that members of the organization would be motivated by their value of safety to pay close attention to hazard-related information.

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 ([1] and [23]).

The discipline of organizational culture derives from an anthropological tradition of studying culture and organizations. It examines patterns of meaning, values, and frames of reference 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 difficult 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 [31]. A similar model suggests that individual behavior is influenced by the triad of organizational structure, organizational processes, and organizational culture [27]. 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 about safety. Thus, safety climate measured in surveys is a manifestation of some aspect of the organizational safety culture. Even with 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 [27]. 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 [32].

1. One question inquires about the organizational culture as an attribute of the organization—as something the organization has. Measurement methods appropriate to this question include observation and audits.
2. A second question asks how the organizational culture impacts individual attitudes about safety. Measurement methods for the second question include surveys and observation.

3. A third question inquires about 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 the reason (purpose) for measuring it.

See [25] for opinions about incident reporting, managers, 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 [32] 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 on the following four factors:

1. Individual communication competence (Section F.4.1).
2. Participatory communication climate (Section F.4.2)
3. Cross-disciplinary or interdisciplinary competence (Section F.4.3)
4. 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 self-image, relationships and tasks as follows [33]:

1. Present a competent and credible image of self.
2. Escalate, maintain, or terminate relationships.
3. Accomplish instrumental tasks.

The research about group communication and interdisciplinary communication indicates that sole focus on tasks, ignoring self-image and relationships, increases the hazard space and prevents organizations from developing collective mindfulness. Thus, the assumption that competent communicators manage the triple {self-image; relationships; tasks} pervades the subsequent discussions.

One commonly cited model of communication competence identifies six factors of communication competence, measurable with a survey [33]:

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 nonverbal 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 principles stated in Section F.1 and develop collective mindfulness. There are four characteristics of participatory communication climate, measured with a survey published in [34], that contribute to collective mindfulness:

1. Individuals have voice to express ideas and concerns.
2. The organization has an open communication climate.
3. Individuals have easy access to relevant information.
4. Individuals engage in continuous and ongoing learning.

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 [9][10][34]. Unfortunately, interdisciplinary communication is also challenging.

In particular, the following communication activities are contribute to hazardous conditions, because they limit organizations' ability to develop interdisciplinary competence [35]:

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.

These behaviors might seem minutiae, but in excess might limit an organization's ability to seek and use intellectual diversity for recognizing hazards.

Teams can increase intellectual diversity by developing collective competence in interdisciplinary group communication. The following behaviors increase collective communication competence [35]:

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 a sense of shared meaning.

Through these behaviors, individuals can manage the triple {self-image; relationships; tasks} mentioned in 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 [36]. Groupthink occurs when group members feel an undesirably strong sense of cohesiveness.

F.4.4.1 Characteristics of Groupthink

Following are some of the identified characteristics of groupthink [36][37]:

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[147] that can improve conversational quality for hazard analysis have been identified ([38] through [40]):

1. Carefully gather information to identify a hazard and 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 might include:
   2.1. Make premises and assumptions explicit [41];
   2.2. Prevent the diffusion of accountability in the organization (see Section F.2); and
   2.3. Measure the group's conversational quality using the Competent Group Communicator Scale [37].
3. Identify factors to reduce the hazard space and seek a range of constraint alternatives.
4. Critically evaluate the identified hazard (from act 1) and the alternatives to reduce the hazard space (from act 3).
5. Select the best course of action to pursue in order to avoid, eliminate, or otherwise control the hazard; remain open to new information; and be willing to change course as needed.

F.5 Collective mindfulness and competence

Survey measures of collective mindfulness are in [34] and [42].

[147] These five conversational acts have been modified to adjust them 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, and (5) select the best proposal.

F.6 References for Appendix F

[1] Bierly III, P.E., and J.-C. Spender, "Culture and High Reliability Organizations: The Case of the Nuclear Submarine," *Journal of Management* 21(4):639–656, August 1995.

[2] Hofmann, D.A., R. Jacobs, and F. Landy, "High Reliability Process Industries: Individual, Micro, and Macro Organizational Influences on Safety Performance," *Journal of Safety Research* 26(3):131–149, Autumn 1995.

[3] Rijpma, J.A., "Complexity, Tight-Coupling and Reliability: Connecting Normal Accidents Theory and High Reliability Theory," *Journal of Contingencies and Crisis Management* 5(1):15–23, March 1997.

[4] Roberts, K.H., D.M. Rousseau, and T.R. La Porte, "The Culture of High Reliability: Quantitative and Qualitative Assessment Aboard Nuclear-Powered Aircraft Carriers," *The Journal of High Technology Management Research* 5(1):141–161, Spring 1994.

[5] Skjerve, A.B., "The Use of Mindful Safety Practices at Norwegian Petroleum Installations," *Safety Science* 46(6):1002–1015, July 2008.

[6] Weick, K.E., K.M. Sutcliffe, and D. Obstfeld, "Organizing for High Reliability: Processes of Collective Mindfulness," in Boin, A., ed., *Crisis Management,* Volume III, Thousand Oaks, CA: Sage, 2008, pp. 31–66.

[7] Schulman, P.R., "The Negotiated Order of Organizational Reliability," *Administration & Society* 25(3):353–372, November 1993.

[8] Swanson, E.B., and N.C. Ramiller, "Innovating Mindfully with Information Technology," *MIS Quarterly* 28(4):553–583, December 2004.

[9] Weick, K.E., and K.M. Sutcliffe, *Managing the Unexpected: Resilient Performance in an Age of Uncertainty*, Hoboken, NJ: Jossey-Bass, 2007.

[10] Weick, K.E., and K.H. Roberts, "Collective Mind in Organizations: Heedful Interrelating on Flight Decks," *Administrative Science Quarterly* 38(3):357–381, September 1993.

[11] Sanders, S., "A Failure of Imagination?" entry at "Wit and Wisdom of an Engineer," available at http://witandwisdomofanengineer.blogspot.com/2011/05/failure-of-imagination.html.

[12] Cheney, G., et al., *Organizational Communication in an Age of Globalization: Issues, Reflections, Practices*, 2nd edition, Long Grove, IL: Waveland Press, June 2010.

[13] Reiman, T., and P. Oedewald, "Assessment of Complex Sociotechnical Systems - Theoretical Issues Concerning the Use of Organizational Culture and Organizational Core Task Concepts," *Safety Science* 45(7):745–768, August 2007.

[14] Pidgeon, N., and M. O'Leary, "Man-Made Disasters: Why Technology and Organizations (Sometimes) Fail," *Safety Science* 34(1–3):15–30, February 2000.

[15] Simon, H.A., *Administrative Behavior: A Study of Decision-Making Processes in Administrative Organizations*, 4th edition, New York: Free Press, 1997.

[16] Toulmin, S., *The Uses of Argument*, Cambridge, UK: Cambridge University Press, 1958.

[17] Bracewell, R.H., et al., "Capturing Design Rationale," *Computer-Aided Design* 41(3):173–186, March 2009.

[18] Eng, N.L., et al., "More Space to Think: Eight Years of Visual Support for Rationale Capture, Creativity and Knowledge Management in Aerospace Engineering," *Proceedings*

of the 23rd International Conference on Design Theory and Methodology and the 16th Design for Manufacturing and the Life Cycle Conference, Washington, DC, August 28–31 2011, ASME, New York, NY, pp. 225–235.

[19] Conklin, J., *Dialogue Mapping: Building Shared Understanding of Wicked Problems*, Hoboken, NJ: John Wiley & Sons, January 2006.

[20] Perrow, C., *Normal Accidents: Living with High-Risk Technologies*, Updated Edition, Princeton, NJ: Princeton University Press, September 1999.

[21] Vaughn, D., The Challenger Launch Decision: Risky Technology, Culture, and Deviance at NASA, Chicago, IL: University of Chicago Press, January 1996.

[22] Perin, C., Shouldering Risks: The Culture of Control in the Nuclear Power Industry, Princeton, NJ: Princeton University Press, October 2006.

[23] Pidgeon, N., "The Limits to Safety ? Culture, Politics, Learning and Man-Made Disasters," *Journal of Contingencies and Crisis Management* 5(1):1–14, March 1997.

[24] International Nuclear Safety Advisory Group (INSAG), "Basic Safety Principles for Nuclear Power Plants," 75-INSAG-3, International Atomic Energy Agency, Vienna, Austria, 1988.

[25] Antonsen, S., "The Relationship between Culture and Safety on Offshore Supply Vessels," *Safety Science* 47(8):1118–1128, October 2009.

[26] Choudhry, R.M., D. Fang, and S. Mohamed, "The Nature of Safety Culture: A Survey of the State-of-the-Art," *Safety Science* 45(10):–, 2007.

[27] Schein, E.H., *Organizational Culture and Leadership*, 4th Edition, Hoboken, NJ: Jossey-Bass, August 2010.

[28] Weick, K.E., "Organizational Culture as a Source of High Reliability," *California Management Review* 29(2):112–127, Winter 1987.

[29] Guldenmund, F.W., "The Use of Questionnaires in Safety Culture Research - An Evaluation," *Safety Science* 45(6):723–743, July 2007.

[30] Pidgeon, N., "Safety Culture: Key Theoretical Issues," *Work & Stress* 12(3):202–216, July–September 1998.

[31] Geller, E.S., "Ten Principles for Achieving a Total Safety Culture," *Professional Safety*, September 1994, pp. 18–24.

[32] Cox, S.J., and A.J.T. Cheyne, "Assessing Safety Culture in Offshore Environments," *Safety Science* 34(1–3):111–129, February 2000.

[33] Canary, D.J., M.J. Cody, and V.L. Manusov, *Interpersonal Communication: A Goals-Based Approach*, Third Edition, Boston: Bedford/St. Martin's, January 2003.

[34] Novak, J.M., and T.L. Sellnow, "Reducing Organizational Risk through Participatory Communication," *Journal of Applied Communication Research* 37(4):349–373, 2009.

[35] Thompson, J.L., "Building Collective Communication Competence in Interdisciplinary Research Teams," *Journal of Applied Communication Research* 37(3):278–297, 2009.

[36] Janis, I.L., *Crucial Decisions: Leadership in Policymaking and Crisis Management*, New York: The Free Press, 1989.

[37] Beebe, S.A., and J.T. Masterson, *Communicating in Small Groups: Principles and Practices*, 8th edition, Boston: Allyn & Bacon, 2004.

[38] Orlitzky, M., and R.Y. Hirokawa, "To 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* 32(3):313–341, June 2001.

[39] Gouran, D.S., and R.Y. Hirokawa, "Effective Decision Making and Problem Solving: A Functional Perspective," in Hirokawa, R. Y., et al., eds., *Small Group Communication: Theory and Practice: An Anthology*, 8th edition, Los Angeles: Roxbury, 2003.

[40] Wittenbaum, G.M., et al., "The Functional Perspective as a Lens for Understanding Groups," *Small Group Research* 35(1):17–43, February 2004.

[41] Antonsen, S., K. Skarholt, and A.J. Ringstad, "The Role of Standardization in Safety Management - A Case Study of a Major Oil & Gas Company," *Safety Science* 50(10):2001–2009, December 2012.

[42] Barrett, M.S., et al., "Validating the High Reliability Organization Perception Scale," *Communication Research Reports* 23(2):111–118, June 2006.

APPENDIX G: An Example Case Study

This case study illustrates that 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-and-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 in which 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 [1] in such generalization have identified two types of reasoning processes, “abduction” and “induction.”

G.1 Ft. Calhoun Event

The following information is based on [2][3].

The plant was shut down on April 9, 2011, for a refueling outage. The outage was extended because of flooding along the Missouri River. Then an electrical fire on June 7, 2011, led to the declaration of an “Alert” and caused further restart complications.

The fire 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 (EQ) analyses for plant structures, systems, and components. Figure 13 illustrates the causality relationships extracted from this textual information. The bottom two blocks illustrate a generalization from the specific occurrence at Ft. Calhoun. In this example, the deficiency in the component design was not caught during the verification and validation (V&V) activities.

These analyses (e.g., the EQ analysis shown in the lowest block of Figure 13) are relied on to demonstrate that key systems will be able to perform their safety functions under a variety of challenging accident conditions such as earthquakes, loss-of-coolant accidents, high radiation fields, seismic events, etc. This generalization is shown in the block in the upper right corner of Figure 13.

To extend the generalization further, among the known causes of “deficient designs,” the leading cause is “deficient requirements.” In the context of RIL-1101, “deficient requirements including cosntraints” results from inadequate HA; for example:

- Inadequate understanding of contributory hazards.
- Inadequate formulation of requirements to avoid or prevent such contributory hazards.
- Inadequate validation of the HA and the resulting requirements.

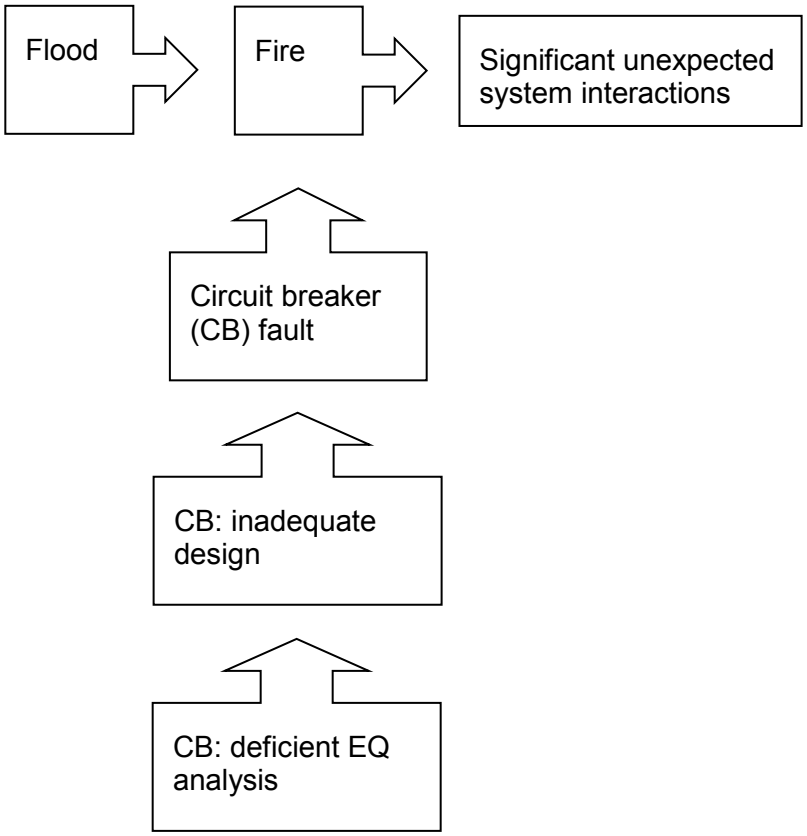

Figure 13: Example from event on June 7, 2011, at Ft Calhoun nuclear power plant (NPP).

"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*, Chapter 4, 1792. [4]

G.2 References for Appendix G

[1] Flach, P.A., and A.M. Hadjiantonis, eds., *Abduction and Induction: Essays on Their Relation and Integration*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 2000.

[2] Omaha Public Power District, "Inadequate Flooding Protection Due to Ineffective Oversight," Licensee Event Report 285-2011-003, May 1, 2011.

[3] U.S. Nuclear Regulatory Commission, NRC, "Fort Calhoun Station - NRC Followup Inspection - Inspection Report 05000285/201007; Preliminary Substantial Finding," IR 05000285-10-007, July 15, 2010, Agencywide Documents Access and Management System (ADAMS) Accession No. ML101970547.

[4] Dictionary.com, "The power of generalizing ideas of, Mary Wollstonecraft," available at http://quotes.dictionary.com/The_power_of_generalizing_ideas_of_drawing_comprehensive.

APPENDIX H: Examples of NPP Modes

Following are examples of modes of a nuclear power plant (NPP) across its lifecycle to serve as reminder that each might present different kinds of hazardous conditions:

1. Construction
2. Preoperational
3. Startup testing
4. Commissioning
5. Operational
6. Testing or maintenance being performed
   6.1. Setpoint adjustment
   6.2. Instrument calibration
   6.3. Change (switching) of calibration parameters (in common position (CP) 2.1.3.2.5 in [1])
7. Refueling or open vessel (for maintenance)
   7.1. Refueling or open vessel—all or some fuel inside the core
   7.2. Refueling or open vessel—all fuel outside the core
8. Decommissioning

Following are examples of modes of an operational nuclear power plant (NPP) to serve as reminder that each might present different kinds of hazardous conditions:

1. Start-up
2. On Power
   2.1. Raising power
   2.2. Full allowable power
   2.3. Reducing power
   2.4. Reduced power (including zero power)
3. Hot Shutdown (reactor subcritical)
   3.1. Hot standby (coolant at normal operating temperature)
   3.2. Hot shutdown (coolant below normal operating temperature)
4. Cold Shutdown (reactor subcritical and coolant temperature < 93°C)
   4.1. Cold shutdown with closed reactor vessel
   4.2. Mid-loop operation – applies to pressurized-water reactor (PWR) only

Reference

[1] Task Force for Safety Critical Software, “Licensing of Safety Critical Software for Nuclear Reactors: Common Position of Seven European Nuclear Regulators and Authorised Technical Support Organizations,” Revision 2013, available at http://www.hse.gov.uk/nuclear/software.pdf.

APPENDIX I: Evaluation of Timing Analysis

Author: Dr. John Stankovic, University of Virginia
http://www.cs.virginia.edu/people/faculty/stankovic.html

Integrative editing by Sushil Birla

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 reanalyzed 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 –four are listed below and are elaborated in the subsequent sections.

1. First (Sections I.1 and I.2) is a complete and explicit layout of all tasks on timelines 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 (central processing units (CPUs), field-programmable gate arrays (FPGAs), etc.), the assignment of tasks to each processing element, and message slots on buses and their purpose.
2. 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).
3. Another approach might be to use dynamic priorities and apply their associated analysis (Section I.3.2). This approach is less deterministic, but has advantages in many situations and can be used as an offline analysis to guarantee timing.
4. A fourth approach is use of FPGAs (Section I.4). In all the design approaches, realistic estimated times should be identified. Accounting for redundancy and fault-tolerance techniques in the design must be included.

I.1 Timing analysis by hand

The developer, using a manual approach, may present a set of timelines with all tasks assigned deterministically. How they created these time lines (possibly by hand) might not be known and is generally very complex. For the evaluator, once the deterministic timelines are given, it is much simpler to check (one by one) whether the set of assignments and timelines 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 because any change at all results in a complete recreation 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 timelines and assignments using some algorithm or heuristics implemented as a computer program. The evaluator would analyze the resulting schedules as in Section I.1. This approach is more desirable than in Section I.1, because changes can be more easily handled than having to recreate schedules and timelines 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 an underlying operating system (OS) executes tasks based on priority.

I.3.1 Mathematical analysis of timing with fixed priorities

Rate Monotonic Analysis (RMA) [2] is a set of techniques to assign fixed priorities and perform an associated timing guarantee analysis. RMA has been used successfully in some avionics systems and in control systems in automobiles.

RMA focuses on periodic tasks, explained in the next paragraph, but can be extended to address both periodic and aperiodic tasks. RMA can incorporate the complexities mentioned 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%, it is guaranteed that all deadlines will be met. This is true even though there are preemptions.

A periodic task T of period p means that an instance of that task is activated every time interval p. Once a particular instance of the task is activated, it has its own deadline d. When the activated instance's d is p time units from its activation time, then RMA applies. This type of real-time task is analyzed using this timing requirement and is commonly referred to as a periodic task with the assumption that deadlines equals periods. If, for each of the particular activations of task T, d is less than p time units from its activation time, then RMA does not apply and a new analysis called Deadline Monotonic Analysis (DMA) is required [2].

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, EDF and its analysis [3] can be applied offline and timing guarantees are possible. In this case, the results are very similar to the fixed-priority approach except that the OS is running an EDF scheduler instead of a fixed-priority scheduler. An evaluator might also see EDF as a basis for the “timing analysis by a program” approach mentioned in Section I.2.

I.4 FPGAs

Various functions in the system may be implemented in hardware (today typically through an FPGA) [5]. Then execution speed of the function can be greater than on a CPU. Functions implemented on an FPGA can be considered tasks in the overall timing analysis and can be

considered to be subject to the analysis techniques[148] described in this appendix. Of course, issues such as input/output (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 might 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 caused by interrupts. See Chapter 5 in [2].

I.5.2 Resources

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 RMA this is handled by the priority ceiling version of RMA; see pages 5-47 through 5-60 in [2]. For EDF, see Chapter 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, analog-to-digital (AD) conversion must execute, the result must then be sent to a processor, a task must execute to process the data, and the processed result must then be 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 RMA- or EDF-based approaches. See pages 3-10 and 3-11 in [2] and Chapter 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 when execution of the tasks is finished or possibly at any point within the execution of the task). Controlling jitter is often necessary for I/O. See Chapter 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 RMA (Chapter 6 in [2]).

[148] National Instruments is an example of a source of tools currently available for use in common practice (for example, the LabView development system together with the Real-Time Module and FPGA module).

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. 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 is 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 might not account for the time it takes to select and switch between tasks. This would be incorrect. See pages 392 through 395 in [4].

I.6.4 Richard's Anomalies

Scheduling can lead to hazardous conditions subtly. For example, if a set of timelines 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), the previous schedules which worked (i.e., with all deadlines met) might now miss deadlines even though individual tasks are executing more quickly. There are four variations of these anomalies (see pages 42 through 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 one is not expected, is important. Will the system fail safely (that is, be fail safe)? Could a catastrophic cascade of deadline misses be caused by 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.

I.8 References for Appendix I

[1] Kopetz, H., *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Second Edition, Berlin: Springer, April 2011.

[2] Klein, M., et al., *A Practitioner's Handbook for Real-Time Analysis*, 1994 edition, Dordrecht, The Netherlands: Kluwer Academic Publishers.

[3] Stankovic, J., et al., *Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 1998.

[4] Buttazzo, G., *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Third Edition, Berlin: Springer, September 2011.

[5] National Instruments, "NI LabVIEW FPGA Module," available at http://sine.ni.com/nips/cds/view/p/lang/en/nid/11834.

[6] Feiler, P.H., and D.P. Gluch, *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language*, Boston: Addison-Wesley, September 2012.

APPENDIX J: Assumptions

Author: Dr. John Stankovic, University of Virginia
http://www.cs.virginia.edu/people/faculty/stankovic.html

Integrative editing by Sushil Birla

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 might become invalid over time. It is also common that combinations of assumptions might cause failure or contribute to hazards. For example, a component (hardware or software) might be 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., in requirements, design and analysis, implementation, and 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) and 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. Statements of assumptions may be classified in any of three ways:

1. Formal-like languages: For example, in AADL [1] an assumption can be stated with an ***assumes*** keyword and some condition 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 categorized by type (e.g., see Table 27) and incorporated in 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 such a statement 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)	Backup power is available 24 x 7.
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 a 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 might 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 the control algorithm is correct and in time.

J.2 Monitoring an assumption at run time

Because 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 runtime 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] Feiler, P.H., D.P. Cluch, and J.J. Hudak, “The Architecture Analysis &Design Language (AADL): An Introduction,” Technical Note CMU/SEI-2006-TN-011, Software Engineering Institute, Pittsburgh, PA, February 2006, available at http://www.sei.cmu.edu/reports/06tn011.pdf.

[2] Lewis, G.A., T. Mahatham, and L. Wrage, “Assumptions Management in Software Development,” Technical Note CMU/SEI-2004-TN-021, Software Engineering Institute, Pittsburgh, PA, August 2004, available at http://www.sei.cmu.edu/reports/04tn021.pdf.

[3] The Eclipse Foundation, “Eclipse - The Eclipse Foundation open source community website,” available at http://www.eclipse.org/home/index.php.

[4] SysML.org, “SysML.org: SysML Open Source Specification Project,” available at http://www.sysml.org/.

[5] University of California–Berkeley, “Ptolemy Project Home Page,” available at http://ptolemy.eecs.berkeley.edu/.

APPENDIX K: Dependency

Authors:

Dr. John Stankovic, University of Virginia http://www.cs.virginia.edu/people/faculty/stankovic.html

Dr. 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

K.1 Purpose and scope

This appendix explains the term, dependency, as it is used in RIL-1101.

In software it is often noted that if module A *uses* module B, then module A depends on module 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-and-effect relationship.

A number of dependencies exist within developed systems and 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 a nuclear power plant (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[149] in a requirement.
- Defect or deficiency in the implementation of the application software.
- Defect or deficiency in the 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 that depend on process transients.

Item 3 in Section 2.4.2 of RIL-1101 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.

[149] Issue: If requirements are deficient, the terms “failure” and “defect” are not applicable; the common-cause failure (CCF) notion, applied to a specified system, does not serve well; and 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. A contributor may influence a hazard in many ways (paths or channels or couplings);[150] for example:

1. Function.
2. Control flow.
3. Data or Information.
4. Sharing of resources.
5. Constraint on resources; for example: Explicit preference order.
6. Conflicting goals or losses of concern.
7. States or conditions in the environment.
   7.1. Controlled processes.
   7.2. Supporting physical processes.
8. Fault.
9. Constraints.
10. Assumptions.
11. Concept.
12. Some unintended, unrecognized form of coupling.

K.4 Examples of dependencies

Dependencies exist within and across hardware and software components and also result from 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 from the physical world.

1. **Sensors**: Software signal-processing and decisionmaking algorithms depend 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**: The power needed to run the actuator and the accuracy of applying command signal influence the output.
3. **Central processing units (CPUs) and memories**: Speed of the CPU, implementation features such as caches and branch prediction, size of memory, type and location of memories on buses, and power requirements influence the output.
4. **Field-programmable gate arrays (FPGAs)**: Speed, power, timing, and availability of inputs influence the output.
5. **Buses**: Communication between distributed devices and software depends on the bus speed and access protocols; it might also depend on a hierarchy of buses.
6. **I/O devices**: Speeds, power, locations, and read and write techniques influence their outputs.
7. **Physical properties**: Sizes of sensors, actuators, and computing devices; I/O interconnection types; temperatures produced by devices; reliability of devices; fault models;

[150] 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.

and the question of whether the system will degrade over time without renewal or maintenance (a form of entropy) influence the consistency of the system output.

8. **Time**: Guaranteeing deadlines depends on the time requirements of real-world phenomena, the speed of hardware, the software processing required, and scheduling algorithms; delays can accumulate.
9. **Location**: Placement of sensors, actuators, and displays influence the system output.
10. **Environment**: External conditions such as earthquakes, hurricanes, power outages, humidity, and fire influence the system output.
11. **Control**: The accuracy of models under which control algorithms were created and the availability or maximum delay of inputs to controller influence the output.
12. **Chain of events**: A particular series of events influences the system output.
13. **Humans**: Reaction time, awareness, and expertise influence their output.

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 relationships between and among uses.
3. **Runtime environment**: The operating system (OS), its version, and particular settings (configurations) and algorithms being used constitute the runtime environment. It is necessary to ensure that unexpected modules are not being run, e.g., modules for system monitoring or periodic cleanup which are accounted for.
4. **Resources**: Amount of CPU time, memory, and bus bandwidth.
5. **Name**: Components are assumed to be named consistently.
6. **Data**: Location, synchronization, availability, and redundancy.
7. **Ordering**: Some sets of components must run in a strict or partial order.
8. **Race conditions**: If some condition causes uncertainty in the time of completion of some functions or the time of arrival of some data and thus the order of these occurrences, it is called a race condition. If the order of occurrence can affect the value of the system output it could be a hazard – sometimes known as a race hazard.

The following examples demonstrate how tight specifications, assumptions, and constraints interrelate logically and might 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, A and B, for data values in a system are in a dependency if, given the value of A, the value of B is affected by the value of A (e.g., fixed to a specific value or 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 depend on a function implemented in hardware. An example would be:

“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, denoted as **common_cause** A, B has the following meaning:
if there is an event C for which the conditions

A **depends_on** C

and

B **depends_on** C

hold.

K.4.4 Example of a resource dependency

These different types of dependencies may interact. For instance, a resource dependency might cause a functional dependency. Two functions, A and B, that are intended to be independent but use the same resources can unintentionally become dependent. If function A might compromise the shared resource in a certain situation in such a way that function B is no longer available, and vice versa. this is a bidirectional dependency between A and B. Then, the hazard analysis (HA) of the system should include the analysis of this dependency.

K.4.5 Dependency through assumptions and constraints

Constraints on interactions can cause dependencies.

- Properties of the environment might actually be assumptions (example: “The water temperature cannot change by more than 10 degrees within 10 milliseconds”).
- Properties of system elements might interact with such assumptions (example: “Whenever the temperature changes by more than 1 degree, the sensor issues a signal”).
- In this way, dependencies are created by constraints on the 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 - see Appendix J.

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, the dependency "(P3) is a logical consequence of (P1) and (P2)" is of relevance for the hazard analysis. If (P1) or (P2) are changed, the conclusion of the hazard analysis might 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. If this is the case, 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 represents logical dependencies between logical statements (in terms of logical entities) using the mathematical relation, "logical implication." Given a number of logical propositions, their implication relationships can be combined applying deduction rules. This related set is a subnetwork of the complete network of logical dependencies in the system. This subnetwork can lead to proof trees (see [1]).

K.5 Dependencies can network

For a system of the kind in RIL-1101's focus, dependencies are not simple chains or trees, but a network (also known as directed graph or digraph [2]); for example:

- The same factor might recur in many places in the network (i.e., common causes might exist).
- 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 2.4) or to its element (Sections 2.7 through 2.9). It can also be applied to the technical processes (Section 2.3), for developing a safety-related system or its element. It can also be applied to the organizational processes (Section 2.2) that influence the development processes.

K.6 Dependencies can propagate through faults

Many dependencies also exist for faults in systems. Hazard analysis should include the analysis of the dependencies across faults to find out whether a fault can propagate and degrade a

safety function. This requires a fault-propagation specification and component fault behavior specification, an explicit specification of fault types propagated, and an explicit specification of system fault states [3][151].

K.7 Unrecognized dependency

Missing, wrong, unwanted, or misunderstood dependencies might contribute to a hazard. If A can have an unwanted effect on B, then B is in some sense dependent on A. In other words, B is not independent of 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 $\mathcal{E}$ might not be determinable. For example, consider the effects of resource sharing and of a memory leak.

There are so many sources of unwanted dependencies that it is easy[152] to miss one. As soon as one is discovered or suspected, it should be documented. Once this is done, known methods can be applied to perform the analysis.

Unrecognized dependencies are defects in hazard analysis and might lead to degradation of a safety function.

For complicated dependencies, many observations are needed to uncover dependency [4].

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
- system timing

This documentation can be made very explicit (for example, in proposition P1, "event A leads to event B," and proposition P2, "event B leads to event C") or it can be implicit in such a way that a dependency can be concluded from explicit stated dependencies (for 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 respectively), 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 formalized way for events A and B in a system:

A **depends_on** B

[151] This reference uses the term "error," which is mapped into the term "fault" in RIL-1101.

[152] In current practice.

This proposition expresses that there is some causal relationship between A and B. Such a causal relationship can have many different aspects and implications:

- A cannot happen before B has happened; as an 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 in which a sensor measures the pressure and sends the values to the alarm manager (event B).
- A is guaranteed to happen if B has happened; as an 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 in which a sensor measures the pressure and sends the values to the alarm manager (event B); an instance of “incorrect ‘pressure too high’ data measured at sensor” (event B) leads to an incorrect alarm (event A).
- A cannot happen if B has happened; as an 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 in which a sensor that measures the pressure and sends the values to the alarm manager over a communication line, but 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, as a result of every behavior, event A may interfere with B; it means that in some instance of behavior, A does interfere with B.

Note furthermore that the relationship
A **depends_on** B

Is not symmetric, in general, and is not even transitive. The same holds for its negation
A **is_independent_of** B

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. Because 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.

Similar to the formulation of a formal predicate characterizing dependencies between events, discussed above, relationships can be characterized between data attributes in states and, more generally, one can formulate rules for dependencies in data and control flow and the propagation of their effects.

System dependencies can be reflected in system models. The models should contain enough information to understand dependencies and propagation paths for contributory hazards (see the suggestion in Note 1 to Table 22 in Appendix C.5 for 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. For example, the work product might be a model of requirements, model of architecture, or 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 (which is, therefore, not discoverable through analysis).

The system's 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). 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 [5]. HA should utilize this information including rules to deduce further dependencies from explicitly documented ones.

Thus, a well-specified architecture is essential for dependency analysis (see [3] and [5]).

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[153] dependencies include the following: Call graphs, IDLs [6], data flow diagrams, and design languages (graphical or not) such as AADL [7] 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.

-Traditional

[153] These are not necessarily complete and are only as good as the information recorded in them.

K.12 References for Appendix K

[1] Broy, M. “A Logical Approach to Systems Engineering Artifacts and Traceability: From Requirements to Functional and Architectural Views,” in Broy, M., D. Peled, and G. Kalus, eds., *Engineering Dependable Software Systems*, Amsterdam: IOS Press, 2013, pp. 1–48.

[2] Garrett, C., and G. Apostolakis, “Context in the risk assessment of digital systems,” *Risk Analysis* 19(1):23–32, February 1999.

[3] Delange, J., and P. Feiler, “Supporting Safety Evaluation Process using AADL,” *Proceedings of the 7th Layered Assurance Workshop, New Orleans, LA, December 9–10, 2013*, Applied Computer Security Associates, Silver Spring, MD, available at http://www.acsac.org/2013/workshops/law/2013-law-proceedings.pdf.

[4] Pfaltz, J.L., “Logical Implication and Causal Dependency,” available at http://www.cs.virginia.edu/~jlp/06.ICCS.pdf.

[5] Feiler, P., and A. Rugina, “Dependability Modeling with the Architecture Analysis & Design Language (AADL),” CMU/SEI-2007-TN-043, Software Engineering Institute, Pittsburgh, PA, July 2007, available at http://www.sei.cmu.edu/reports/07tn043.pdf.

[6] Object Management Group, Inc., “OMG IDL,” available at http://www.omg.org/gettingstarted/omg_idl.htm.

[7] Feiler, P.H., D.P. Cluch, and J.J. Hudak, “The Architecture Analysis & Design Language (AADL): An Introduction,” Technical Note CMU/SEI-2006-TN-011, Software Engineering Institute, Pittsburgh, PA, February 2006, available at http://www.sei.cmu.edu/reports/06tn011.pdf.

[8] SysML.org, “SysML.org: SysML Open Source Specification Project,” available at http://www.sysml.org/.