# The Product Engineering Class in the Software Safety Risk Taxonomy for Building Safety-Critical Systems

Janice Hill
*NASA, KSC, Florida*
*Florida Institute of Technology, Melbourne, Florida*
*Janice.L.Hill@nasa.gov*

Daniel Victor
*ManTech SRS, KSC, Florida*
*Florida Institute of Technology, Melbourne, Florida*
*Daniel.C.Victor@nasa.gov*

## Abstract

*When software safety requirements are imposed on legacy safety-critical systems, retrospective safety cases need to be formulated as part of recertifying the systems for further use and risks must be documented and managed to give confidence for reusing the systems. The SEI Software Development Risk Taxonomy [4] focuses on general software development issues. It does not, however, cover all the safety risks. The Software Safety Risk Taxonomy [8] was developed which provides a construct for eliciting and categorizing software safety risks in a straightforward manner. In this paper, we present extended work on the taxonomy for safety that incorporates the additional issues inherent in the development and maintenance of safety-critical systems with software. An instrument called a Software Safety Risk Taxonomy Based Questionnaire (TBQ) is generated containing questions addressing each safety attribute in the Software Safety Risk Taxonomy. Software safety risks are surfaced using the new TBQ and then analyzed. In this paper we give the definitions for the specialized Product Engineering Class within the Software Safety Risk Taxonomy. At the end of the paper, we present the tool known as the 'Legacy Systems Risk Database Tool' that is used to collect and analyze the data required to show traceability to a particular safety standard.*

## 1. Introduction

Governmental agencies and industry often require the use of safety standards in contracted projects to ensure that the systems produced are completed in an ordered manner. The combination of design, analysis, inspection, and test activities, when consistently performed throughout the system development lifecycle, has shown to be extremely successful, as in the International Space Station flight computer system.

In general, safety standards and requirements are required to be in place at the beginning of a program or project, but this is not always the case. The NASA Software Safety Standard is one such standard that contains process-oriented software safety requirements that are to be met by all NASA developed or contracted safety-critical software, regardless of its age. The standard states that each NASA legacy system should be assessed for the software's contribution to the safety of the system, and then planning should commence for the individual legacy system to meet or not meet the requirements of the standard. The NASA 'software safety litmus test' is used for the assessment of the software in the system for safety criticality. A common approach to aid in the determination of whether or not a safety-critical system meets software safety requirements was not specified in the NASA software safety standard. For this reason the Software Safety Risk Taxonomy was proposed as a framework to assess risk in legacy safety-critical computer systems when attempting to apply the software safety standard after the fact.

Gauging *software safety risk* [8] is an essential part of determining the specific activities and depth of analyses needed to meet software safety requirements. The implementation and approach to meeting software safety requirements will vary to reflect the system to which they are applied. [1]

Performance of safety standards is part of what is called making a 'safety case'. A safety case is the documented demonstration that the system complies with the specified safety requirements. [2] To provide safety assurance, evidence needs to be gathered on the integrity of the system and put forward as an argued

case, e.g., the safety case, that the system is adequately safe. [2] A safety case is not a new concept, however making a formal safety case has not been required for NASA's legacy safety-critical systems. The usefulness of safety and dependability cases is being investigated for some new development projects within NASA for certifying systems as safe and secure. The authors believe that it is equally important to make safety cases for legacy systems that may be reused in support of new major programs and projects.

Problems occur when attempting to fulfill the requirements of a software safety standard in a legacy real-time safety-critical computer system. In the past, researchers have investigated the problem of retrospectively making a safety case for the software, to meet new safety standards in the industry. [3] The risk of not meeting certain software safety requirements is a topic that needs to be addressed when attempting to make safety cases for legacy safety-critical computer systems. This is also a reason for devising a new taxonomy specifically designed to facilitate the identification of software safety risks.

This paper presents the definitions for the Product Engineering Class and the corresponding safety elements and attributes that belong to the Software Safety Risk Taxonomy. The taxonomy was proposed in a earlier paper [8] and is based on the Software Development Risk Taxonomy created and used by the Software Engineering Institute (SEI) [4, 5, 6], with an additional Legacy element added to the Product Engineering class, as in the Risk Taxonomy authored by Batista Webster *et al.* [7] The SEI taxonomy was originally chosen as the model for the Software Safety Risk Taxonomy because it maps very well to the structure of the NASA Software Safety Standard. The NASA-STD-8719.13B addresses not only new software development, but also legacy, heritage and reused safety-critical software systems.

A Software Risk Evaluation (SRE) is a practice that was developed by the SEI containing a formal method for identifying, analyzing, communicating and mitigating software technical risk. [5] The SEI Software Development Risk Taxonomy is a part of this practice. In our research we are using the Software Safety Risk Taxonomy in addition to the SEI Taxonomy to generate a comprehensive list of questions for defining an inclusive set of risks for legacy safety-critical computer systems. The Software Safety Risk Taxonomy addresses the additional safety related tasks and analyses that are required over and above traditional software engineering process activities.

A pilot study using the SEI taxonomy and the original Taxonomy Based Questionnaire (TBQ) was initiated using one of the NASA legacy systems at the Kennedy Space Center (KSC) in Florida. This system monitors and controls ground support equipment at the launch site, facility power and the sound suppression water system. Preliminary data was collected and risks were captured. The next step in the pilot is to use the new Software Safety Risk Taxonomy and its corresponding TBQ. The Software Safety Risk TBQ will be used to interview participants in a similar fashion to elicit the software safety risks. Results obtained from the interview process will be used when following the rest of the SRE practice. Once the pilot study is complete, the forward plan is to work with 4-6 other legacy systems at both KSC and Wallops Flight Facility in Virginia.

## 2. Problems with Developing Retrospective Safety Cases

It can be assumed that not all safety requirements in a safety standard can be met for a legacy system and therefore a software safety risk assessment must be performed. The assumption that an existing system is safe may not hold when the legacy system is used in a new application.

There are three basic problems with developing retrospective safety cases. The first is reliance on the safe use of the system over the years, and the objective evidence of the safety issues. Safety issues include reporting and analysis of accidents, incidents and resultant problem reports. The second problem is with attempting to show that the design of the legacy system is acceptable in the present, even though it was developed to standards current at the time. In addition to this, the legacy system may not have been designed to any standard at all, and this will also be a consideration when creating a safety case. The third problem is that of missing information. Information is lost over time; vendors go out of business or are no longer under contract with the maintainer of the system. [9]

With legacy systems, it can be a difficult task to construct a safety case, because there may be few to no artifacts available to show compliance with the software safety requirements. Because of this, there will be risks associated with not meeting safety requirements in a legacy safety-critical system. These software safety risks must be addressed by project management to give confidence for reusing an existing system. Risk factors in general will be different for legacy safety-critical computer systems, and the software within them. Knowing the risks, project managers can then decide whether to try to recreate missing artifacts or accept the risks of not having certain safety documents or analyses to make the safety case. This is another reason for a taxonomy

specifically focused on identifying software safety risk factors.

## 3. The Software Safety Risk Taxonomy

The Software Safety Risk Taxonomy, like the SEI taxonomy, maps the characteristics of *safety-critical* software development, and therefore of *safety-critical* software development risks. [4] The Software Safety Risk TBQ consists of questions only related to those additional safety activities essential to producing and maintaining safety-critical software. Figures 1, 2 and 3 illustrate the three classes and their elements and attributes of the safety taxonomy. [8]

A. Product Engineering

1. *Safety* Requirements
   - *Identifiable*
   - Stability
   - Completeness
   - Clarity
   - Validity
   - Feasibility
   - *Safety requirements traceability*
   - *Safety requirements analysis*
2. *Safety* Design
   - *Safety* Functionality
   - Difficulty
   - *Safety* Interfaces
   - *Safety* Performance
   - *Safety* Testability
   - Hardware Constraints
   - Non-Developmental Software
   - *Safety design traceability*
   - *Safety design analysis*
3. *Safety* Code and Unit Test
   - Feasibility
   - *Safety* Testing
   - Coding/Implementation
   - *Safety code traceability*
   - *Safety code analysis*
4. *Safety* Integration and Test
   - *Safety* Environment
   - Product Integration
   - *Safety test traceability*
   - *Safety test analysis*
5. Engineering Specialties
   - *Safety* Maintainability
   - Reliability
   - Security
   - Human Factors
   - Specifications
6. *Legacy*
   - *Reverse engineering*
   - *Replacement*

**Figure 1. Product engineering class**

The Product Engineering class in Figure 1 is the largest class in the Software Safety Risk Taxonomy so we will address it in this paper. It contains the elements numbered 1 through 6 that cover the development of the safety-critical system products. Under each element are the attributes of interest for eliciting safety risks.

B. Development Environment

7. *Safety* Management Process
   - *Safety* Planning
   - *Safety* Organization
   - *Safety* Management Experience
   - *Safety* Program Interfaces
8. *Safety* Management Methods
   - *Safety* Monitoring
   - *Safety* Personnel
   - *Safety Assurance*
   - *Safety* Configuration Management
9. Work Environment
   - *Safety* Attitude
   - Cooperation
   - Communication
   - Morale

**Figure 2. Development environment class**

The Development Environment class in Figure 2 covers the project environment and the process used to engineer safety-critical system products. [8] It contains the elements numbered 7 through 9 in the taxonomy.

C. Program Constraints

10. *Safety* Resources
    - *Safety* Schedule
    - *Safety* Staff
    - *Safety* Budget
    - *Safety* Facilities

**Figure 3. Program constraints class**

The Program Constraints class in Figure 3 contains the factors that may be outside of the control of the project responsible for the safety-critical system development. [8] Its sole element is numbered 10 in the taxonomy.

In section 4 we provide the descriptions of the Product Engineering taxonomic class, elements and attributes which are specialized for safety. In section 5 are some sample questions. The 'Legacy Systems Risk Database Tool' that is being developed as part of this research project, is briefly described in section 6.

## 4. Software Safety Risk Taxonomy Product Engineering Class

Product engineering is defined as the technical processes to define, design and construct or assemble a product. [11] Product engineering for safety is defined as the technical processes used to build a safety-critical product. It refers to the system engineering and software engineering activities involved in creating a safety-critical system that satisfies specified safety requirements and customer expectations. [4] Activities include system hazard analysis, system and software safety requirements analysis and specification, system and software safety design and implementation, integration of hardware and software components, and software and system test for safety-critical systems.

The elements of this class cover the safety engineering activities that are necessary to be performed over and above traditional system and software engineering activities.

In the Product Engineering class the software safety risks that will most likely be generated will relate to inadequate analysis of the system for the technical software safety requirements. Additionally, software safety risks can also be linked to insufficient safety design features. Coding standards that do not include safety considerations might also contribute to software safety risks. [8]

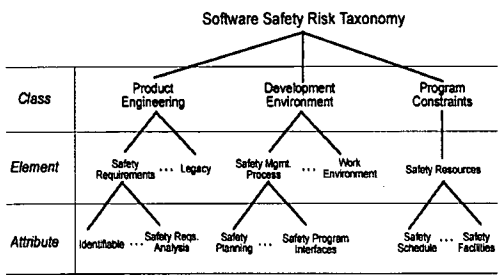


**Figure 4. Software safety risk taxonomy**

Figure 4 shows the high level schematic of the software safety taxonomy, and the relationships between the Product Engineering class and the remaining classes.

### 4.1. Safety Requirements

A requirement is defined as a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. [11] Safety requirements are defined as 1) process oriented requirements e.g., what needs to be done to ensure software safety and 2) technical requirements that specify what the system must include or implement. [1] The attributes of the safety requirements element include the quality of the safety requirements and the complexity of the implementation that satisfies the safety requirements. Safety requirements may be insufficient or missing if system hazard analyses did not include the software in the analysis or assumed that the software works as expected all the time.

#### Identifiable

Identifiable is defined as capable of being identified. [12] The identifiable attribute refers to the ability to identify in order to track the safety requirements based on some unique classification or identification scheme.

#### Stability

Stability is defined as having a marked tendency to remain unchanged. [12] The stability attribute refers to the degree to which the safety requirements are changing and the possible effect that changing safety requirements and external interfaces will have on the quality, safety, functionality, schedule, design, integration and testing of the product being built.

#### Completeness

Completeness is defined as having all the necessary parts, elements or steps. [12] The completeness attribute refers to missing or incompletely specified safety requirements. Requirements documents that do not include the safety requirements, or do not adequately specify the safety requirements, or have safety requirements that are "to be defined", or have been inadvertently omitted, will result in lack of budget for safety requirements.

#### Clarity

Clarity is defined as the quality or state of being clear [12] e.g., absence of ambiguity. The clarity attribute refers to ambiguously or imprecisely written safety requirements. If the customer and provider do not have a mutual understanding of the safety requirements, then there may be safety requirements rework later in the development cycle when it is more costly to fix.

**Validity**

Validity is defined as being well-grounded or justifiable; logically correct. [12] The validity attribute refers to whether or not the safety requirements in total reflect the customer's expectation for the safety of the delivered system. This attribute may be affected by ambiguity in safety requirements, unwritten customer expectations or constraints, or a specification where not all the stakeholders had a chance to make inputs to the safety requirements.

**Feasibility**

Feasibility is defined as the degree to which the requirements, design or plans for a system or component can be implemented under existing constraints. [11] The feasibility attribute refers to the difficulty of implementing a single technical safety requirement or of meeting two or more safety requirements that may conflict when implemented together in the system. Additionally, feasibility means the ability to decide on a tolerable qualification method for demonstrating that the system satisfies the safety requirements.

**Safety Requirements Traceability**

Traceability is defined as the ability to trace the history, application or location of an entity by means of recorded identifications. [1] Safety requirements traceability is defined as the ability to show the source of a particular safety requirement and the linkage from that source, in this case the hazard report, to the safety requirement and back to the source. The source of the safety requirements should be known and available to start the traceability for the rest of the safety design and development of the system. This includes both technical and process-oriented safety requirements. The software development/management plans should be reviewed for inclusion of the process-oriented software safety requirements.

**Safety Requirements Analysis**

Requirements analysis is defined as the process of studying and refining system, hardware or software requirements. [11] Safety requirements analysis is defined as the process of studying and refining the safety requirements that result from system hazard analyses. This attribute refers to the activities that occur early in the life of the system. The concept of 'what the system shall do' is used to perform system level preliminary hazard analysis. Hazards and the risk associated with those hazards are identified. Fault tree analysis is performed using the identified hazards, which will include the contribution of the software in the system. Safety requirements are derived and assigned to hardware and software parts of the system.

## 4.2. Safety Design

Design is defined as the process of defining the architecture, components, interfaces, and other characteristics of a system or component, as well as the result of this process. [11] Safety design is defined as the features and methods e.g., inhibits failure detection and recovery, interlocks, assertions and partitions that are incorporated in the software design. [1] The attributes of the safety design element include algorithms that are designed for minimum risk or include 'fail operational/fail safe' or 'single fault tolerant' or 'two fault tolerant' requirements; safety functional and performance requirements, internal and external safety interfaces. The following attributes characterize the safety design element.

**Safety Functionality**

Functionality is defined as the particular set of functions or capabilities associated with computer software or hardware or an electronic device. [12] Safety functionality is defined as the functions provided by the safety-critical software. The safety functionality attribute covers the safety functional requirements that may not be designed sufficiently to meet minimum risk, fault tolerance or fail operational/fail safe requirements. This includes the algorithms or designs that may not meet the overall system safety requirements.

**Difficulty**

Difficulty is defined as the quality or state of being hard to do, make or carry out. [12] The difficulty attribute here refers to safety functional or design requirements that may be difficult to achieve. The system architecture as designed may be difficult to implement to meet the design for minimum risk requirements.

**Safety Interfaces**

Interfaces are defined as hardware or software components that connect two or more other components for the purpose of passing information from one to the other. [11] Safety interfaces are defined as the hardware or software components specifically designed as interfaces to safety-critical software, or that implement safety requirements. The

safety interface attribute covers all hardware and safety critical software interfaces and the techniques for defining and managing the interfaces. This includes commercial off the shelf (COTS), government off the shelf (GOTS), modified off the shelf (MOTS) and legacy-heritage-reused software and developmental hardware interfaces.

**Safety Performance**

Performance is defined as the degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage. [11] Safety performance is defined as the ability of a safety-critical system to handle periodic capacity, load and timing requirements; this is a fundamental safety property. [10] The safety performance attribute refers to time critical performance; real time response requirements, performance analyses, reliability analyses, user response requirements, 'must work' and 'must not work' requirements, failure detection, isolation and recovery requirements.

**Safety Testability**

Testability is defined as the degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met. [11] Safety testability is defined as the ability of a safety design to be tested to meet safety criteria. It refers to the design of the safety features to facilitate testing and the inclusion of safety personnel in the design process to facilitate the development and performance of safety tests.

**Hardware Constraints**

Hardware constraint is defined as the requirement for, or restriction or limitation to using specific hardware in a system. In a safety context, it refers to the system and processor architecture required to meet the system and software safety requirements. The constraints may include memory size, throughput, real-time response capability, database access or capacity limitations, computer hardware type such as firmware, Programmable Logic Controller (PLC), Field Programmable Gate Array (FPGA), or personal computer usage versus 'big iron' mainframe usage.

**Non-Developmental Software**

Non-developmental software is defined as software that is 1) developed in-house such as government-off-the-shelf, 2) not developed in-house such as off-the-shelf, 3) software developed for a different project other than the current project it is being used for, such as reused software. [10] The Non-Developmental Software - NDS (COTS, GOTS, and MOTS, legacy-heritage-reused) attribute refers to the risks with system requirements that may not quite meet the system and software safety requirements. The customer may not accept vendor development, test, or reliability data that would demonstrate satisfaction of system and software safety requirements for the NDS. It may be difficult to show the 'pedigree' of the NDS when safety certification of the system is required.

**Safety Design Traceability**

Safety design traceability is defined as the ability to show the source of a safety design feature and the linkage from that source, in this case, the safety requirements, and back to the source. Since the source of the safety design is the safety requirements, this constitutes a continuation of the traceability that was started in the requirements phase. The process documents, such as software design documents, are reviewed for the inclusion of process-oriented software safety requirements.

**Safety Design Analysis**

Design analysis is defined as the process of studying and refining system, hardware or software designs. Safety design analysis is defined as the process of studying and refining the safety design features and methods. It refers to the activities that occur during the software design phase such as criticality analysis, risk assessments, and independence analysis. The software design is analyzed for areas or conditions that may lead to further hazards being created. The fault tree that was started in the requirements phase is updated as a result of the software design activities.

## 4.3. Safety Code and Unit Test

Code is defined as computer instructions and data definitions expressed in a form suitable for input into a compiler or translator. [11] Safety code and unit test is defined as the safety-critical software in a system and the test process that is performed on the individual safety-critical code units. It refers the safety-critical software implementation e.g., safety-critical code, safety interface specifications and constraints.

**Feasibility**

Feasibility is defined as the degree to which the requirements, design, or plans for a system or component can be implemented under existing constraints. [11] The feasibility attribute for safety of the code and unit test element refers to problems that may be created as a result of poor safety design.

**Safety Testing**

Unit test is defined as testing of individual hardware or software units or groups of related units. [11] Safety testing is defined as test activities for safety-critical units to verify functional software safety requirements. [1] It refers to the planning for unit test for safety-critical functions and the resources and time for the test activities. Planned test cases especially designed for safety-critical units, code units that have been peer reviewed, safety simulations and hardware necessary to accomplish the test plan are required.

**Coding/Implementation**

The coding/implementation attribute with regards to safety covers language constraints, coding standards, development and target hardware constraints specific to safety-critical systems.

**Safety Code Traceability**

Safety code traceability is defined as the ability to show the source of a safety feature in the code and the linkage from that source, in this case the safety design, and back to the source. Since the source of the safety feature in the code is the safety design, this constitutes a continuation of the traceability that was started in the requirements phase and continued in the design phase. The process documents, such as coding standards, are reviewed for the inclusion of process-oriented software safety requirements.

**Safety Code Analysis**

Code analysis or inspection is defined as a static analysis technique that relies on visual examination of development products to detect errors, violations of development standards, and other problems. [11] Safety code analysis is defined as the process of reviewing safety-critical code for errors or defects. It refers to the activities that occur during the software coding/implementation phase. Computer software units designated as safety-critical are reviewed for correct and complete safety requirements implementation. The code is also reviewed for contributions to hazards. The fault tree that was updated in the design phase is further refined as a result of the software coding/implementation activities.

## 4.4. Safety Integration and Test

Integration is defined as the process of combining software components, hardware components, or both into an overall system. [11] Test is defined as an activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component. [11] Safety integration and test is defined as the integration of the safety-critical software and hardware and the test process that is performed on the integrated system. It refers to the integration and test planning, execution and facilities required for the safety-critical development products and the safety-critical system.

**Safety Environment**

A test bed is defined as an environment containing the hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test. [11] The safety environment is defined as the integration and test environment that is equipped to represent the safety capabilities of the operational environment.

**Product Integration**

The product integration attribute with regards to safety refers to the integration of the safety-critical software components and the hardware, and subsequent testing of the integrated system. Safety interfaces, testability of safety requirements, adequacy of test plans for safety, regression testing for changes, and ample time and resources for integration and test are factors to consider.

**Safety Test Traceability**

Safety test traceability is defined as the ability to show the source of a safety testing requirement and the linkage from that source, in this case, the safety critical code, and back to the source. Since the source of the safety testing is the safety-critical code, this constitutes a continuation of the traceability that was started in the requirements phase and continued in the coding/implementation phase. The process documents, such as test plans and procedures, are reviewed for the inclusion of process-oriented software safety requirements.

### Safety Test Analysis

Test analysis is defined as analyses performed before the fact to ensure validity of the tests, and analyses of the test results. [10] Safety test analysis is defined as reviewing the methods and results of testing and documenting and reporting any improperly implemented safety features. [1] It refers to the activities that occur during the software integration/test phase. Problem reports, safety verification matrices and test reports are reviewed. The fault tree that was updated in the code/implementation phase is further refined as a result of the integration and testing activities.

## 4.5. Engineering Specialties

The engineering specialties are defined as other quality attributes that complement the safety attributes.

### Safety Maintainability

Maintainability is defined as the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment. [11] Safety maintainability is defined as how well the safety-critical system was planned and executed to meet the technical and process-oriented safety requirements and how difficult it is to make changes. Safety maintainability may be weakened by not following safety standards or safety processes.

### Reliability

Reliability is defined as the ability of a system or component to perform its required functions under stated conditions for a specified period of time. [11] NASA defines software reliability as the discipline of software assurance that (1) defines the requirements for software controlled system fault/failure detection, isolation, and recovery; (2) reviews the software development processes and products for software error prevention and/or reduced functionality states; and (3) defines the process for measuring and analyzing defects and defines/derives the reliability and maintainability factors. [1] The reliability attribute as it relates to a safety critical system involves the degree of control, complexity and timing criticality of the software part of the system. These characteristics have a strong influence on the development of safe and reliable software. [10]

### Security

Security is defined as a discipline focusing on preventing unauthorized access to classified information and preventing malicious activities. [13] The security attribute as it relates to a safety-critical system is the ability for unauthorized access into software part of the system.

### Human Factors

Human factors. is defined as an applied science concerned with designing and arranging things people use so that the people and things interact most efficiently and safely. [12] The human factors attribute as it relates to a safety-critical system involves analyzing the potential human errors in the system, tasks that should be done by the system and those performed by humans, and the policies and management that should be in place to develop the system safely.

### Specifications

A specification is defined as a document that specifies, in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristics of a system or component, and, often, the procedures for determining whether these provisions have been satisfied. [11] This attribute addresses the safety specifications for the system. These specifications may be formal specifications, functional or at the program level. Safety specifications will convey the technical safety requirements, commonly referred to as computer based control system requirements and should be well documented and controlled.

## 4.6. Legacy

Legacy as it relates to legacy systems is defined as an old computer system or application program that continues to be used because the user (typically and organization) does not to replace or redesign it. [14] The attributes of the legacy element address the activities required to determine the safety requirements that were implemented and safety processes employed in the legacy system.

### Reverse Engineering

Reverse engineering is defined as to disassemble and examine or analyze in detail (as a product or device) to discover the concepts involved in manufacture usually in order to produce something similar. [12] The reverse engineering attribute refers to

the various methods and tools used to produce documentation required by safety standards and safety requirements. This attribute addresses the difficulty of implementing safety standards after the fact if these standards are imposed on legacy safety-critical systems.

**Replacement**

Replacement is defined as one that replaces another especially in a job or function. [12] The replacement attribute refers to the activity of replacing all or part of a legacy safety-critical system. Replacement can result in risks of finding equivalent safety-critical systems.

## 5. Software Safety Risk Taxonomy Based Questionnaire (TBQ)

Figure 5 below outlines some sample questions from the Software Safety Risk Taxonomy Based Questionnaire. These are representative of the questions used to formulate software safety risks in the Product Engineering class, safety requirements element, and safety requirements analysis attribute.

**A. Product Engineering**

**1. Safety Requirements**

h. Safety requirements analysis
*[Are safety requirements analyzed using a specified methodology?]*

[1] Was a Preliminary Hazard Analysis (PHA) performed for this system?
(Yes) Is the PHA available for review?
(Yes) Is software included as a part of the PHA?

[2] Was a System Safety Analysis (SSA) performed for this system?
(Yes) Is the SSA available for review?
(Yes) Is software included as a part of the SSA?

[3] Are the system and software safety requirements analyzed for proper flow down from the system level requirements?
(No) Who is responsible for doing the safety analyses?

[4] What types of safety analyses are performed?
a. Requirements Criticality Analysis
b. Software Fault Tree Analysis
c. Software Safety Requirements Flow-down Analysis
d. Timing, Throughput and Sizing Analysis
e. Peer Reviews and Inspections of safety requirements
f. Traceability Analysis
g. Control Flow Analysis
h. Information Flow Analysis

[5] Are safety analyses documented?
(Yes) Are the documented analyses results under configuration control?

**Figure 5. Sample taxonomy based questions**

## 6. The Legacy Systems Risk Database Tool

In addition to the development of the Software Safety Risk Taxonomy, this research project includes the design and construction of a database tool. The Legacy Systems Risk Database Tool is used to 1) load safety standard requirements into the database, 2) automate the collection of data, which includes both the software development and software safety risks, 2) provide a reporting capability, including software safety risk metrics reports, 3) provide a decision making mechanism for project management, and 4) create and show traceability links from the risks to the selected safety standard requirements.

Figure 6 below shows a subset of the data model for the database tool. The diagram portrays the relationships between the Software Safety Risk Taxonomy, the TBQ, the software safety requirements, the answers to the questions, and the software safety risks, for each project.

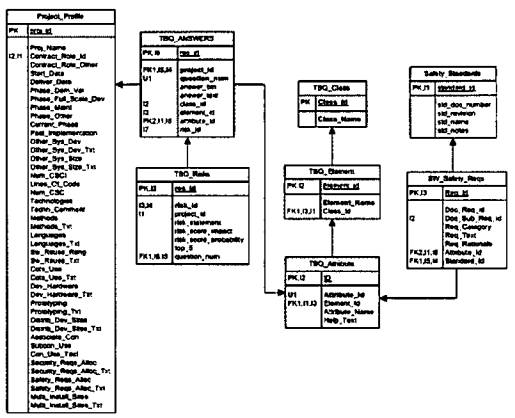

**Figure 6. Legacy systems risk database architecture**

Figure 7 shows a screenshot of the prototype TBQ data entry user interface. There will be several more user interfaces developed once all of the safety related questions are developed based on the new safety taxonomy.

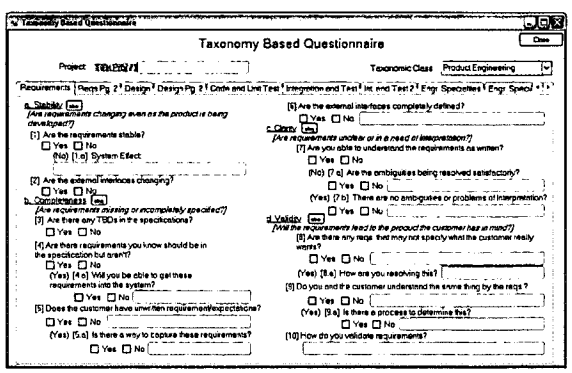


**Figure 7. Legacy systems risk database prototype user interface**

## 7. Summary

This paper describes the Product Engineering Class for the Software Safety Risk Taxonomy that will enable a retrospective safety case to be made for legacy safety-critical computer systems being considered for reuse. In another follow-on paper we will describe the Development Environment and Program Constraints classes to complete the Software Safety Risk Taxonomy. Subsequently, the Software Safety Risk Taxonomy Based Questionnaire (TBQ) will be constructed that contains the safety related questions to elicit software safety risks. The Legacy Systems Risk Database Tool that is currently in development will serve as a tracking, control and communication mechanism for the software safety risks that are uncovered by the Software Safety Risk TBQ.

## 8. Acknowledgement

This research is part of a project funded by the NASA Headquarters Office of Safety and Mission Assurance and the NASA IV&V Facility to investigate 'Assurance and Recertification of Safety-Critical Software in Legacy Systems". A three year research proposal was submitted in July 2006 to the NASA Headquarters Office of Safety and Mission Assurance (OSMA) to address assurance problems for the safety of legacy systems and to identify practical software safety techniques.

## 9. References

[1] NASA Office of Safety and Mission Assurance, *NASA-STD-8719.13B Software Safety Standard w/Change 1*, 2004.

[2] S. Gardiner (ed.) *"Testing Safety-Related Software, A Practical Handbook"*, Springer-Verlag, London, 1999.

[3] T.M. Bull, E.J. Younger, K.H. Bennett, Z. Luo, "Bylands: Reverse Engineering Safety-Critical Systems", IEEE, 1995.

[4] M. J.Carr, S. L. Konda, I. Monarch, F. C. Ulrich, C. F. Walker, "Taxonomy-Based Risk Identification", *Software Engineering Institute Technical Report, CMU/SEI-93-TR-6*, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1993.

[5] G.J. Pandelios, S.G. Behrens, R. L. Murphy, R.C. Williams, and W.R. Wilson, "Software Risk Evaluation (SRE) Team Member's Notebook (Version 2.0), *Software Engineering Institute Technical Report, CMU/SEI-99-TR-029*, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1999.

[6] R. P. Higuera, Y. Y. Haimes, "Software Risk Management" *Software Engineering Institute Technical Report, CMU/SEI-96-TR-012*, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1996.

[7] K. P. Batista Webster, K. M. de Oliveira, N. Anquetil "A Risk Taxonomy Proposal for Software Maintenance", *Proceedings of the 21st IEEE International Conference on Software Maintenance,* (ICSM '05), IEEE Computer Society, 2005.

[8] J. Hill "A Software Safety Risk Taxonomy for Use in Retrospective Safety Cases", *Proceedings of the 31st Annual IEEE/NASA Software Engineering Workshop*, (SEW '07), IEEE Computer Society, 2007.

[9] A.J. Shears, and T. Cockram, "An e-Safety Case Approach to Assuring Safety in UK Legacy Air Launched Munitions", Retrieved January 8, 2007 from, http://www.praxis-cs.com/eSafetyCase/downloads/parari_paperv2.pdf

[10] NASA Office of Safety and Mission Assurance, *NASA-GB-8719.13 NASA Software Safety Guidebook*, 2004.

[11] Standards Coordinating Committee of the Computer Society of the IEEE, *IEEE Std. 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology*,The Institute of Electrical and Electronics Engineers, New York, 1990

[12] http://www.m-w.com/dictionary, Retrieved January 8, 2007 from, Merriam-Webster, Incorporated, 2007-2008

[13]N.G.Leveson,*"Safeware System Safety and Computers"*, Addison-Wesley, Boston, 1995 , pp. 183.

[14] http://en.wikipedia.org/wiki, Retrieved January 8, 2007 from, *Wikipedia, The Free Encyclopedia*, Wikimedia Foundation, Inc., 2007

ASWEC Special Sessions

# Issues in ICT Education

This session is concerned with ICT education from the holistic perspective of preparation in high schools, the university experience, transition to the workforce, and the contribution by industry, government, and professional bodies.

Consultations to date with various stakeholders have revealed numerous concerns that include: the dispersed nature of the ICT sector; erroneous perceptions of ICT disciplines; decline in enrolments; gender imbalance; lack of industry involvement; and balancing knowledge with generic skills acquisition.

This forum will involve the project team and invited participants to report on findings from consultations and research. Attendees will be invited to respond and to contribute their issues and challenges that are of particular concern to the software engineering community.

Professor Joe Chicharo

Project Director

Professor Fazel Naghdy

Project Director

Dr Tony Koppi

Project Manager

# Aerospace and Real Time Issues

This special session addresses crucial issues in a variety of applications particularly in the aerospace industries and real time control systems, risk assessment, software reliability, software safety, error protection, reuse for reliability and safety, as well as issues in software development. These issues are of importance to a large number of other industries such as Offshore Oil and Gas Platforms and remote Resource Industry sites. We have put together a selection of papers that address different parts of these themes. The session will take the form of a lead presentation of 30 minutes by the session chair followed by 10 minute position paper presentations of each of their concerns followed by a 40 minute open panel discussion with audience participation and panel participation.

This Special session is organized and chaired by research scientists from the National Aeronautics and Space Administration, NASA, Kennedy Space Center, Florida, USA. NASA's mission is to pioneer the future of space exploration, scientific discovery, and aeronautics research. The Chairs are Computer Engineers performing research focusing on Safety of Computer Based Control Systems.

Janice Hill

Daniel Victor