A Software Safety Risk Taxonomy for Use in Retrospective Safety Cases

Janice Hill
NASA, Kennedy Space Center, Florida
University of Florida, Gainesville, Florida
Janice.L.Hill@nasa.gov

Abstract

Safety standards contain technical and process-oriented safety requirements. The best time to include these requirements is early in the development lifecycle of the system. When software safety requirements are levied on a legacy system after the fact, a retrospective safety case will need to be constructed for the software in the system. This can be a difficult task because there may be few to no artifacts available to show compliance to the software safety requirements. The risks associated with not meeting safety requirements in a legacy safety-critical computer system must be addressed to give confidence for reuse. This paper introduces a proposal for a software safety risk taxonomy for legacy safety-critical computer systems, by specializing the Software Engineering Institute's 'Software Development Risk Taxonomy' with safety elements and attributes.

1. Introduction

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

Government agencies and industry often choose to impose the use of safety standards, which include software safety requirements, in projects to ensure that the systems produced are done so in a structured manner. The combination of analysis, inspection, design and test activities, when consistently performed throughout the system development lifecycle, has shown to be extremely successful, as in the Space Shuttle on-board flight control computer system.

There are many safety standards to choose from, some directed at the system level and others intended only for use in the creation of software. In general, safety standards are meant to be employed in the beginning of a project as opposed to later. Implementation 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 problem arises when attempting to fulfill the requirements of a software safety standard in a legacy real-time safety-critical computer system. In the Bylands research project [3], Bull, *et al.* poses the question, "how do we retrospectively make a safety case for the software, perhaps to meet new safety standards in the industry?" This project focused on an analysis of the software code for its safety properties rather than on the safety process followed to develop the software.

The software safety risk taxonomy proposed here is based on the Software Development Risk Taxonomy developed 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 chosen as a basis for the software safety risk taxonomy because it maps very well to the structure of the NASA Software Safety Standard, which is being used in the author's research project. The NASA-STD-8719.13B contains software safety requirements for both new and legacy systems.

A Software Risk Evaluation (SRE) is an important component of the decision process inherent in making a safety case. The SRE practice that was developed by the SEI is a formal method for identifying, analyzing, communicating and mitigating software technical risk. [5] For the purposes of this paper, the portions of the SEI's SRE methodology of interest are: 1) a proposed software safety taxonomy, similar in content to the

SEI's taxonomy, enabling the definition of risk factors specific to legacy safety-critical software, and 2) the resulting set of questions focused on eliciting software safety risks.

Several legacy systems located at the NASA Kennedy Space Center will be selected for investigating the feasibility of the new taxonomy. The software safety Taxonomy Based Questionnaire (TBQ) will be used to interview participants on the activities and tasks involved with the maintenance and reuse of legacy real-time safety-critical computer systems. Results of the software safety TBQ will be used in the rest of the SRE practice.

2. Safety Case Development

The primary purpose of the safety case is to demonstrate adequate safety in design and operation of systems. Planning and preparing a comprehensive safety case is especially important when developing new real-time safety-critical computer systems and equally important when legacy systems are candidates for reuse in new applications.

Documenting a safety case includes fully describing the safety aspects of the system. Software safety integrity must be considered from the system perspective, and performance standards need to be set with reference to risk reduction to be performed. [4] Gardner [4] presents the some of the evidence needed to justify safety claims that includes:

1. credible arguments about the adequacy of the requirements specifications, in terms of hazards to be addressed
2. credible arguments about the ability of the design and implementation to satisfy the safety requirements
3. credible arguments about the adequacy of the development process in terms of its ability to avoid introduction of faults and to detect and remove faults
4. empirical evidence of the systems ability to satisfy the safety requirements.

2.1. Problems with Developing Retrospective Safety Cases

Safety standards, when levied on a project, are often the basis for making the safety case. Meeting the requirements of safety standards is best accomplished when they are specified in the beginning of the system development. In some instances however, the requirements in a safety standard are levied upon legacy safety-critical computer systems. The NASA Software Safety Standard is one such standard that includes requirements for legacy systems. [1]

When a legacy safety-critical computer system is planned to be reused to monitor and control new hardware, a 'retrospective' safety case must be made. Changes are made to legacy systems to accommodate the differences in the new application, such as advanced technology, new interfaces, and processes. If a safety case is available for the legacy system, it must also be changed.

Not all safety requirements in a safety standard can be met for a legacy system and 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.

Shears and Cockram [9] note three problems with developing retrospective safety cases. The first 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. Additionally, 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 second 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. The third 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.

3. Risk Definitions

Higuera and Haimes [6] write that risk is commonly defined as "a measure of the probability and severity of adverse effects". Software technical risk is "a measure of the probability and severity of adverse effects inherent in the development of software that does not meet its intended functions and performance requirements". [6] Sherer defines software risk as "the expected loss that can occur as software is developed, used or maintained". [8] Each of these definitions contains a common theme that risk is something that is measured. The SEI has found that software risk is among the least measured or managed in a system. [6]

3.1. Software Safety Risk Factors

'Software Safety Risk' is defined in this paper as a measure of the probability and severity of adverse effects inherent in the development of software that does not meet *some set of software safety requirements.* This new definition will be used first in support of the software safety taxonomy when identifying risk factors specific to legacy safety-critical software.

The risk posed by safety-critical software will be calculated based on the software safety risk factors elicited by using the software safety taxonomy. Software safety risk will vary with the system safety criticality, which consists of the type of hazards and the level of control or influence the software has on system safety factors. [1]

In some cases 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. The risks associated with not meeting safety requirements in a legacy safety-critical system must be addressed 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. Thus the need for a taxonomy specifically focused on identifying software safety risk factors.

4. The Software Safety Risk Taxonomy

Carr *et al.* at the SEI developed a taxonomy of risks which focuses on the software development process. [4] Batista Webster *et al.* [7] proposed a taxonomy of risks for software maintenance projects using the SEI taxonomy. The software safety risk taxonomy defined in this paper will be used to produce an in depth set of safety related questions for building a Taxonomy Based Questionnaire (TBQ).

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 TBQ will consist of non-judgmental questions used to bring out issues, concerns and risks in each of the taxonomic classes. [4] Using a structured TBQ will ensure that all risk areas related to the development of safety-critical software are methodically addressed. The overview of the changes and additions to the aforementioned taxonomies are shown in italics in Figure 1, 2 and 3. Definitions for all of the taxonomic groups in the element and attribute categories for the software safety risk taxonomy will be provided in a follow-on paper. The definitions for each of the classes are provided in sections 4.1 through 4.3.

4.1. Safety Elements and Attributes in the Product Engineering Class

In the SEI Software Development Risk Taxonomy, the Product Engineering class contains the system engineering and software engineering activities involved in creating a system that satisfies specified requirements and customer expectations. [5] When a system is identified as safety-critical as a result of a hazard analysis, the software's contribution to the safety of the system is included in the analysis. This contribution necessitates the unique identification of safety requirements that must be traced throughout the software development lifecycle.

For legacy safety-critical computer systems, there will be some investigative work required to discover if and where safety requirements were originally specified for the software. 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. Other software safety risks could be linked to insufficient safety design features. In Figure 1, the additional safety elements and attributes are shown.

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
   - *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 questions for the Product Engineering class will focus on ascertaining what safety and software engineering artifacts and 'tribal knowledge' [10] exists from the original system and software development process.

4.2. Safety Elements and Attributes in the Development Environment Class

The SEI's Development Environment class addresses the project environment and the process used to engineer a software product. [5] The processes, methods and environment for managing a project that produces a safety-critical system will be different. Planning and management for safety-critical systems require documented processes for the development of the safety products such as hazard analyses and software safety analyses. Safety-critical systems development and management not only require the traditional software engineering and support group skills, but also trained system and software safety engineering personnel. Legacy safety-critical computer systems have been found to be lacking in the areas of safety planning and management.

In the Development Environment class the software safety risks that could be elicited are those associated with inadequate planning and management for safety activities in the budget and schedule. Additional risks found could be lack of cooperation or communication between the development and safety organizations. Figure 2 shows how the Development Environment class is modified to include safety process factors.

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 questions for the Development Environment class will center on discovering if any safety planning and management activities existed during the original system and software development process.

4.3. Safety Elements and Attributes in the Program Constraints Class

The Program Constraints class refers to the factors that may be outside of the control of the project responsible for the system development. These factors can still have major effects on the projects success or constitute sources of substantial risk. [5] The development of a safety-critical system necessitates the specific resources and scheduling for the performance of safety tasks and production of the safety products. In the case of a legacy safety-critical computer system, it has been shown that most often there is insufficient or lack of allocation of safety resources to the project.

In the Program Constraints class some software safety risks that may be generated are related to inadequate allocation of budget to obtain skilled safety staff. Some other risks in this class could be concerned with the lack of scheduling the safety resources early enough in the development of the system. Figure 3 identifies the safety elements in the Program Constraints class.

C. Program Constraints

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

Figure 3. Program constraints class

The questions for the Program Constraints class will look for the safety resources that were allocated (if any) during the original system and software development process.

4.4. Using the Software Safety Risk Taxonomy in Retrospective Safety Cases

In general, to make a safety case there must be some objective evidence to prove the safety requirements have been implemented. For legacy safety-critical computer systems, that evidence may or may not exist. To assist project and safety management to make the best judgment as to the integrity of a legacy system where artifacts may be lacking or incomplete, the software safety taxonomy is designed to, 1) provide a framework for organizing data and information, 2) map directly to a well known

software safety standard, and 3) provide a basis for generating questions for discovery of issues, concerns, and risks. The detection of risks is the first function in the Software Risk Evaluation practice. Management can decide based on the software safety risks whether to recreate artifacts, or accept the risks generated by the software safety taxonomy, to make the retrospective safety case.

5. Summary

This paper describes a proposal for a new software safety risk taxonomy that will enable a retrospective safety case to be made for legacy safety-critical computer systems which may be considered for reuse. The taxonomy is based on the SEI's taxonomy of risk factors which focuses on the software development process, and a taxonomy of risk factors for software maintenance projects proposed by Batista Webster *et al.* The software safety taxonomy will be used to generate software safety risk factors and an in depth set of safety related questions for a Taxonomy Based Questionnaire (TBQ). This research is part of a project funded to investigate 'Assurance and Recertification of Safety-Critical Software in Legacy Systems". Results from this initial research will be documented in a follow-on paper.

6. Acknowledgement

This research is supported by the NASA Office of Safety and Mission Assurance (OSMA) and the NASA Independent Verification &Validation (IV&V) Facility, Software Assurance Research Proposal Initiative.

7. References

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

[2] Stewart 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] S. Sherer "The Three Dimensions of Software Risk: Technical, Organizational and Environmental", *Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, IEEE, 1995, pp. 369-380.

[9] A.J. Shears, and T. Cockram, "An e-Safety Case Approach to Assuring Safety in UK Legacy Air Launched Munitions", Retrieved December 29, 2006 from, http://www.praxis-cs.com/eSafetyCase/downloads/parari_paperv2.pdf

[10]http://en.wikipedia.org/wiki/Tribal_knowledge, Retrieved December 29, 2006 from, *Wikipedia, The Free Encyclopedia*, Wikimedia Foundation, Inc., 2006