Software Safety Risk in Legacy Safety-Critical Computer Systems

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

Rhoda Baggs, Ph.D.
Assistant Professor of Computer Sciences
Florida Institute of Technology
rbaggs@fit.edu

Abstract

Safety-critical computer systems must be engineered to meet system and software safety requirements. For legacy safety-critical computer systems, software safety requirements may not have been formally specified during development. When process-oriented software safety requirements are levied on a legacy system after the fact, where software development artifacts don't exist or are incomplete, the question becomes 'how can this be done?' The risks associated with only meeting certain software safety requirements in a legacy safety-critical computer system must be addressed should such systems be selected as candidates for reuse. This paper proposes a method for ascertaining formally, a software safety risk assessment, that provides measurements for software safety for legacy systems which may or may not have a suite of software engineering documentation that is now normally required. It relies upon the NASA Software Safety Standard, risk assessment methods based upon the Taxonomy-Based Questionnaire, and the application of reverse engineering CASE tools to produce original design documents for legacy systems.

1. Introduction

When planning for the production of new software, technical and process-oriented software safety requirements are included during the software development lifecycle. Meeting these types of requirements is part of what is called making a 'safety case'. [3] [4] [5] [9] Process-oriented software safety requirements, such as those contained in government and industry safety standards, may be levied on a legacy system after the fact. Often, software development artifacts don't exist or are incomplete. How can a legacy safety-critical computer system meet these new standards?

The method proposed here is aimed at providing a quantitative measure of the safety of legacy computer systems which are still in use and may be reused but may not have a complete set of software engineering documentation as required by any number of software safety standards. In the NASA "Software Safety Standard", sets of "software safety activities, data, and documentation necessary for the acquisition or development of software in a safety-critical system" are described. [2] As part of the design for a particular system, it is recommended in this NASA standard that "the following documentation shall address safety-critical software". These particular documents are common artifacts of the system and software development processes that should contain all the requirements, methods and procedures for safety-critical software. See Figure 1.

1. Software Safety Plan
2. Software Project Management Plan
3. Software Configuration Management Plan
4. Software Quality Assurance Plan
5. Software Requirements Specification
6. Software Design Documentation
7. Verification and Validation Plan
8. Safety Analyses and Reports
9. Test Documentation
10. User documentation and procedures
11. Operations and Maintenance Plan

Figure 1. NASA Software Safety Standard documentation requirements [2]

Let us call this documentation list the "Beta Set" or simply β.

One plan of action for assessing the safety of a legacy computer system and therefore make a safety case for the software, is to analyze software design documentation artifacts which must either be located or created for legacy systems. Following the methodology, these artifacts will be used or created if necessary to produce such an assessment. At the same time, a new taxonomy catered and tweaked for software safety and based upon work done by the Software Engineering Institute will be used. From this 'software safety risk taxonomy', a Taxonomy-Based Questionnaire (TBQ) is developed and used as part of the formula for quantifying the software safety risk. It is worthwhile to note that this method exhibits the following crucial software engineering characteristics: completeness,

fairness, robustness, and although complex, directly derivable from known technologies.

2. The Problem and Solution

Put simply, a measure of software safety risk must be ascertained for legacy systems in consideration for reuse. General software risk is the expected loss that can occur when software is developed, used or maintained. [1] The risk posed by safety-critical software 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. [2] Calculating software safety risk is an essential part of determining the specific activities and depth of analyses needed to meet process-oriented software safety requirements. It is useful to define a composite function R(F,G) which will give an estimation of the software safety risk involved in using or re-using a particular legacy safety-critical computer system. R(F,G) represents such a function where F is itself a function that calculates a measure of software safety risk on a system with a complete set of software engineering documentation. F is based on the "Taxonomy Based Questionnaire (TBQ)" as outlined in [6] [7] [10]. G is a similar function that calculates the software safety risk based upon documentation supplied by various reverse engineering CASE tools and or techniques.

Once R, F, and G are defined, along with the CASE tools to help produce a software system documentation suite, a calculation of the software safety risk of the legacy system can be made. This results in a set of software safety risk metrics for decision making for mitigation strategies.

3. Formalizing the Problem Solution

Let us define the following formal sets and functions. Note that each will be explained in greater detail in subsequent sections.

S: A legacy safety-critical computer system whose level of software safety based on software safety risk is to be defined.

β: Set of software engineering documentation artifacts which represent software safety requirements for S.

β_1: Set of already existing software engineering documentation for S (worst case β_1 is the empty set, otherwise $\beta_1 \subset \beta$).

β_2: Set of software engineering documentation for S that does not exist and therefore we need to develop using reverse engineering tools (worst case $\beta = \beta_2$, otherwise $\beta_2 \subset \beta$).

F (β): A function (or method or process) for calculating the software safety risk of S, based on β.

G(β_2): A function (or method or process) for calculating the software safety risk of S, based on β_2.

R(F, G): A composite function for measuring software safety risk for S.

F (β) and G(β_2) are mutually exclusive and functionally independent.

4. Defining the Formal Sets and Functions

4.1. The Beta Set (β)

The Beta Set, (β) is either a full or partial subset of the documentation requirements list taken from the NASA Software Safety Standard as listed in Figure 1. This makes β a complete or 'ideal' set of documentation as recommended by NASA for addressing software safety requirements. Ideally, this complete set of artifacts is produced during development. Since this research addresses the problem for legacy safety-critical computer systems, it is more likely that these types of systems will have a partial set. This is especially true for systems and sub-systems from the Space Shuttle and the International Space Station programs, which were initiated prior to the development of the NASA Software Safety Standard. Note that according to the NASA standard, this list can be tailored or specialized for particular systems.

4.2. The Beta One and Beta Two Sets (β_1 and β_2)

β_1 and β_2 represent those software documentation artifacts that do exist for S and those that do not exist for S (β_1 and β_2, respectively).

To define these two sets, first a legacy safety-critical computer system, S is identified as a candidate for reuse. The appropriate project personnel responsible for the legacy system are interviewed to create a project profile, to find out what documentation artifacts exist and if any 'tribal knowledge' of the system is available [15]. Based on the results of the interviews, β_1 is produced. β_2 is then found from $\beta - \beta_1$.

4.3. Calculating F(β)

F(β) is found using the value of β, which is the tailored or specialized set of documents that address the safety-critical software in the candidate legacy computer system, S. F uses the Taxonomy Based Questionnaire since "most project risks are usually known by project personnel and as a consequence can be surfaced and managed" [6]. The proposed function (method or process) that is applied to β is summarized by the following steps:

1. Define a TBQ based on the software safety taxonomy and catered for legacy systems. The software safety taxonomy maps the characteristics of software development of safety-critical software, and hence of software safety risks. [6]

2. Use the TBQ to elicit risks and identify issues (e.g., potential risks). The taxonomy is designed to facilitate the classification of the risks themselves, as associated with the software development process, including the safety requirements [6] [10]. β will map to the taxonomy and so will the risks and issues captured in this process step.
3. Analyze the risk data using methods based on the Software Engineering Institute (SEI) Software Risk Evaluation (SRE) practice. [8] [10] The analysis will aid in translation of the risks for later decision making. [8]
4. Perform risk ranking.

4.3. Calculating $G(\beta_2)$

$G(\beta_2)$ is calculated using β_2, which represents any documentation that wasn't recovered or otherwise obtained with the production of β_1 but has been determined to be useful when the determination of β was made. β_2 is to be supplied by reverse engineering CASE tools and or techniques. The proposed function (method or process), G that is applied to β_2 is summarized by the following steps:
1. Determine appropriate reverse engineering tools for use in developing the specific set of documents defined by β_2.
2. Select and acquire available tools to create the β_2 set, based on step 1.
3. Create the β_2 set of documents. If tools are not available for some of the documents, then the risk of not finding a tool/technique to develop these documents will be assessed as part of function G.
4. Analyze the risk data using methods based on the SEI's SRE practice. [8] [10].
5. Perform risk ranking.

4.4. Calculating R(F, G)

The composite function R(F, G) will be calculated using the results from functions $F(\beta)$ and $G(\beta_2)$. Function R(F, G) has the following steps:
1. Research risk matrices to use for quantifying the complete set of software safety risks calculated so far. Start with investigating the use of the 4x5 risk matrix in the NASA Office of Safety and Mission Assurance Risk Management Procedural Requirements. [13]
2. Determine appropriate risk assessment tools and processes to use with the software safety risks. One example is the DDP tool [14] used at NASA and the Jet Propulsion Laboratory (JPL), which is a risk-informed requirements engineering tool.
3. Select the appropriate risk matrix, risk assessment tools and processes.
4. Use the risk assessment tools and processes to define software safety risk metrics.

During the course of a three year research initiative, it is projected that the composite function R(F,G) will be calculated for several legacy safety-critical computer systems at the Kennedy Space Center, Florida. There are many legacy launch and landing support systems to choose from at this NASA facility, with some currently being considered for reuse to monitor and control the next generation of space vehicles. Thus, the proposed quantitative measure of the software safety risk is expected to assist engineers and management in risk mitigation strategies when these legacy systems are reused.

5. Conclusion

Much published work has been explored with respect to the reverse engineering of software systems, and even with legacy systems, not much has been attempted on the specific issue of software safety risk. The authors believe that the NASA Software Safety Standard, by providing a list of artifacts (Figure 1.) achieves the desired connection proposed here between legacy safety-critical computer systems, and a technique for calculating software safety risk.

This paper outlines a proposal for a methodology that will provide a quantitative measure of the safety of legacy computer systems which are being considered for reuse but may not have a complete set of software engineering documentation as required by software safety standards. The methodology is based on existing risk elicitation and evaluation techniques which will be adapted for software safety requirements. The research project to investigate the methodology is in its early stages. 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 and the NASA IV&V Facility, Software Assurance Research Program.

7. References

[1] 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.

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

[3] L-H.Eriksson, "Using Formal Methods in a Retrospective Safety Case", *SAFECOMP 2004*, LNCS 3219, Springer-Verlag, Heidelberg, 2004, pp. 31-44.

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

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

[6] 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.

[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] 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.

[9] A.J. Shears, and T. Cockram, "An e-Safety Case Approach to Assuring Safety in UK Legacy Air Launched Munitions", Retrieved December 14, 2006 from, http://www.praxis-cs.com/eSafetyCase/downloads/parari_paperv2.pdf

[10] 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.

[11] European Cooperation for Space Standardization Secretariat, "Space Product Assurance Methods and techniques to support the assessment of software dependability and safety", *Draft ECSS-Q-10-80* ESA Publications Division, ESTEC, Netherlands, 2006.

[12] C. Boldyreff, J. Brittle, C. Korhonen, P. Kyaw, J. Lavery, D. Nutter, and S. Rank, "Web-Based Support for Managing Large Collections of Software Artefacts", *Proceedings of the 27th IEEE Annual International Computer Software and Applications Conference* (COMPSAC '03), IEEE Computer Society, 2003.

[13] NASA Office of Safety and Mission Assurance, *NPR 8000.4 Risk Management Procedural Requirements w/Change 1,* Ch. 2, 2004.

[14] M.S. Feather, S.L. Cornford, K.A. Hicks, K.R. Johnson, "Applications of tool support for risk-informed requirements reasoning", *International Journal of Computer Systems Science & Engineering volume 20 no.1,* CRL Publishing Ltd, January 2005.

[15] http://en.wikipedia.org/wiki/Tribal_knowledge, Retrieved December 14, 2006 from, *Wikipedia, The Free Encyclopedia,* Wikimedia Foundation, Inc., 2006.