2010 5th International Conference on System of Systems Engineering

Independent Validation of Software Safety Requirements for Systems of Systems

Stephen B. Driskell
TASC Inc.
1000 Technology Dr.
Fairmont, WV 26554
USA
Stephen.Driskell@TASC.COM

Judy Murphy
MPL Inc.
1000 Technology Dr.
Fairmont, WV 26554
USA
judy_murphy@mail.mpl.com

James Bret Michael*, Man-Tak Shing*
Department of Computer Science
Naval Postgraduate School
1411 Cunningham Rd.
Monterey, CA 93943, USA
{bmichael, shing}@nps.edu

Abstract *– As one can only reason about the safety of software within the system context in which the software is executing, traditional independent validation and verification approaches which focus on the assurance of satisfaction of requirements by the delivered software are not sufficient in the assurance of software safety requirements. This paper presents an extension to an approach to assurance that relies on the use of a system reference model to capture an independent verification and validation team's understanding of a system's requirements. Here we describe how to apply the approach in conjunction with hazard analysis to evaluate the sufficiency of the software safety requirements early in the software development process. We explain this approach using examples of the safe hold requirements for a spacecraft system.*

Keywords: Reuse, Software Safety, Validation

1 Introduction

The National Aeronautics and Space Administration's (NASA) portfolio of major projects ranges from highly complex and sophisticated space transportation vehicles, to robotic probes, to satellites equipped with advanced sensors to study the Earth [1]. Many of these systems function together as complex safety-critical systems of systems (SoS) to support NASA's research missions in science, aeronautics, and human space flight and exploration.

Ensuring that software functions do not contribute to system hazards in an unintended manner is the responsibility of software system safety, which is accomplished via a set of management and engineering activities from the system safety and software engineering domains to identify, analyze, design, and track software mitigation and control of hazards and hazardous functions [2]. As one can only reason about the safety of software within the system context in which the software is executing, the software system safety engineering process typically starts with the system safety engineering activities to identify potential hazards and safety-critical functions, which are then traced through design into safety-critical hardware and software functions. The process ends with validation and verification (V&V) of derived software safety requirements for controlling the hazard causal factors within the engineering design space. (Readers can refer to [3] for definitions of error, failure, mishap, hazard, hazard casual factor and risk.)

To increase the confidence level of the correctness of a SoS, independent verification and validation (IV&V) has been included as a mandatory practice in the aeronautics, space, railway, and defense industries, where dependability is of paramount importance. With this process, a team of software engineers, who are not the members of the development team, is tasked to validate and verify the SoS's software. Traditional IV&V approaches rely mainly on the manual examination of software requirements and design artifacts. They may also analyze the source code, either manually or by employing a software analyzer, and perform independent testing of target code. These approaches focus on the satisfaction of specified requirements by the delivered software and are not sufficient in the IV&V of software safety requirements. We need an approach to independently assure that the specified requirements have sufficiently and adequately mitigated the potential hazards posed by a SoS.

* The research reported in this article was funded in part by a grant from the National Aeronautics and Space Administration. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright annotations thereon.

DOI 10.1109/SOSE.2010.467

It is important for the IV&V team to "formulate its own understanding of the problem and how the proposed system is solving the problem … [because] technical independence ('fresh viewpoint') is an important method to detect subtle errors overlooked by those too close to the solution" [4]. In [5], Drusinsky *et al.* described a System Reference Model (SRM) framework available to the NASA IV&V Program to conduct computer-assisted formal V&V of software systems. The IV&V team uses a set of three questions to describe the necessary behaviors for system-goal satisfaction (i.e., to solve the problem): *What should the system do? What should the system not do? How should the system respond under non-nominal circumstances?* The SRM framework allows the IV&V team to capture its own understanding of the problem and the expected behavior of any proposed system for solving the problem via an executable system reference model, which is made up of a set of use cases, Unified Modeling Language (UML) artifacts (at a minimum, activity, sequence, and object class diagrams), and a set of executable formal assertions to precisely describe the mission- and safety-critical behaviors for satisfying system goals as defined by the answers to the three questions. This paper extends the SRM approach to validate the sufficiency of the software safety requirements of a SoS.

It is becoming a standard practice in system safety to require the developer to provide the certifier or regulator with a safety case, which contains well-documented evidence to provide "a clear, comprehensive and defensible argument that a system is acceptably safe to operate in a particular context" [6]. In [7], Cruickshank *et al.* present a framework for validating software safety requirements that uses a set of metrics to gauge the sufficiency of the software safety requirements. Instead of relying on final testing to reveal any validity issues with the software safety requirements, application of the framework helps to identify potential problems early on in the development lifecycle. While one cannot replace the intellectual process of validating software safety requirements entirely with metrics, the validation method presented in [7] enables the safety engineering team to take a more proactive approach to ensuring the correct software behaviors are identified and implemented.

This paper addresses the need for IV&V to assess the quality of the software safety engineering early in the development of a system or SoS. It builds upon the research presented in [5] and [7] and presents a new approach that allows the IV&V team to capture their understanding of potential hazards and casual factors via the development of a SRM and then use the SRM to evaluate the developer's hazard analysis and fault management requirements.

The organization of the paper is as follows. Section II presents the process for developing a safety case via the construction of a SRM. Section III describes a case study of applying the proposed approach to the validation of the safe hold requirements of a spacecraft in support of a NASA science mission. Section IV concludes with a discussion of future work.

2 Safety Cases

2.1 Modeling of Safety-critical Behaviors

The starting point of both understanding and documenting system behaviors is to identify the high-level use cases (and use case scenarios) from the stakeholder's input, which could be in the form of mission statements, user expectations, and the concept of operations. Properly developed use cases are used to identify the functionality of the system which records the behavioral requirements for the software [8]. The use cases help the system analyst understand the problems to be solved and the objectives to be accomplished by the proposed system software. The high-level use cases are goal-oriented (instead of function-oriented), and typically are used to describe the workflow of a business (or operation) process instead of interactions among system components. Further, mapping the scenarios of the use cases to activity diagrams, sequence diagrams and statecharts helps highlight the assignment of responsibilities among the component systems of a system of systems, the interdependencies among the component systems, and the safety-critical state behavior of the component systems. For example, Figure 1 shows a representative (but fictitious) activity diagram for the fault monitoring and management use case of a spacecraft.

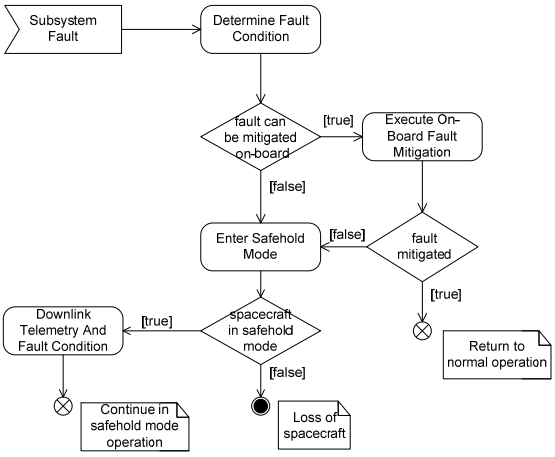

Figure 1 - An activity diagram for fault management

Depending on the risks of the associated safety-critical functions, the high-level use cases must be refined into detailed use cases to capture the second and third level casual factors. Moreover, a use case typically describes what the system should do; the IV&V analysts should also develop misuse cases [9] to understand how the system might be abused or used inappropriately, and to capture what the system should do and should not do under these situations.

The IV&V team then uses the SRM artifacts to identify the potential hazards and generates a list of first- and second-order safety-critical failure events.

2.2 Evaluation of the Developer's Software Safety Products

In [10], Weaver identifies the sufficiency of hazard identification and the adequacy of hazard analysis to identify software's role in causing these hazards as being two important considerations in constructing a safety case. In addition, he also advocates the use of traceability from the derived software safety requirements to the identified system hazards as an indication of the completeness of the set of software safety requirements.

In this paper, we use the terms *sufficiency* and *adequacy* to describe the degree to which discrepancies between the IV&V team's and the project's lists of the potential hazards and the necessary software safety requirements to manage the safety-critical faults exist, which can be assessed by comparing the IV&V team's list of safety-critical failure events against those identified by the project's Failure Modes and Effects Analysis (FMEA) and Fault Management (FM) artifacts. We also examine the traceability between the failures identified in the FMEA and FM artifacts and the traceability of FM functions identified in the developer's system and software requirements artifacts for completeness of the software safety requirements, where the term *completeness* refers to the absence of unmanaged safety-critical faults identified in these artifacts.

3 Safety Case Study

3.1 IV&V Introduction

The mission of NASA's IV&V program, under the auspices of the NASA Office of Safety and Mission Assurance (OSMA), is to provide the highest achievable levels of assurance for mission- and safety-critical software.

The NASA IV&V Program conducted a safety case study for spacecraft safe hold. Safe hold is the autonomous software for managing spacecraft hazards, without ground intervention. It establishes a "safe" stable spacecraft condition, with minimal power consumption, power-positive (sun pointing), battery charging, and omni-directional (ground) communication. Ground operations can recover from the safe event by assessing the spacecraft failure and providing intervention to restore mission operations. This safety case evaluates the sufficiency of software safety requirements involving the use of safe hold to manage hazards that may result in the loss of spacecraft and/or loss of mission. The study resulted in a reusable template for a safety case to ensure that hazards are managed. Mission success and spacecraft safety are both improved through contingency hazard management and the resulting failure risk reduction.

3.2 NASA Mission Assessment

This study provides a process outline for the accomplishment of a safety case for safe hold, which can serve as a template for the evaluation of any future NASA mission.

The NASA IV&V Program used a Portfolio Based Risk Assessment (PBRA) to prioritize and define the system-of-systems analysis scope [11]. The PBRA is a risk matrix which assesses mission capabilities in order to prioritize the development risks for those capabilities. It contains capabilities and rates impact and likelihood of each capability's risks, mapping these into a five-by-five matrix. The following were identified by PBRA as high priority:

- Maintaining the health and safety of the spacecraft which involves execution of and response to faults
- Checkout of the spacecraft which includes safe hold mode and autonomous operations.

The activities that would normally be performed in the accomplishment of the PBRA analysis were leveraged to achieve the following objectives:

- Build a safety case for on-orbit safe hold
- Address the following questions:
  - Does the autonomous action comprehensively manage both the loss of spacecraft and the mission hazards ensuring safety?
  - Are all subsystem faults requiring safe hold included in the safe hold monitor (a safety executive)?
- Ensure these hazards are managed and failure risk is reduced
- Deliver a reusable standardized spacecraft software safety case for IV&V
- Identify missing safe hold requirements
- Provide inputs to software testing

3.3 Hazard Analysis and SRM Development

The study utilized artifacts provided by the satellite developer. A SRM was developed which included high-level use cases, one "Observatory State" state transition diagram, and one "Enter Safe Hold" activity diagram. The study leveraged the existing diagrams and created one new "Fault monitoring and management" activity diagram (similar to the one shown in Figure 1) to capture the context of the "Enter Safe Hold" use case.

We conducted an analysis of the artifacts provided by the satellite developer. Our focus was on safety requirements, fault management, fault responses and failure modes relevant to on-orbit operations; we ignored industrial and ground safety. The following artifacts were reviewed:

- Core Performance Requirements – Level 3[1]
- Core Spacecraft FM
- Core Spacecraft FMEA
- Core Spacecraft Flight Software – Level 4
- Guidance Navigation and Control System Requirements Specification – Level 4
- Core Observatory Command and Data Handling Subsystem Requirements – Level 4
- Software Safety Program Plan (SSPP)
- Preliminary Hazard Analysis (PHA)

The IV&V team created an initial list of first- and second-order failure events that they believe could cause the satellite to enter safe hold. The list includes potential software faults (e.g., out-of-range errors in the Guidance, Navigation and Control calculation) and hardware faults (e.g., power subsystem sensors exceeded thermal limits). This independent list was used to compare the failure data in the developer-provided FM and FMEA artifacts. Gaps were identified in both developer artifacts as well as the list constructed by the IV&V team. The IV&V team then updated the independent list to address the gaps by identifying missing requirements and failure events. Issues and observations are tracked and updated using the latest engineering artifacts as they are delivered. The study is a snapshot in time prepared from engineering work-in-progress.

3.4 Validation of Developer's Safety Requirements

Figure 2 portrays the IV&V analysis process we followed. The IV&V team evaluated the developer-provided artifacts against the OSMA safety criteria. FMEA results in identifying component failures, failure modes, failure mechanism failure effects, failure severity, and recovery mechanisms. FM additionally contains four types of recovery mechanisms: (i) internal automatic recovery, (ii) relative time sequence stored commands (scripts), (iii) processor reset, and (iv) redundant fail-over. Some of these result in spacecraft safe hold. Ground intervention actions are defined to recover from safe hold.

[1] At NASA, Level 1 is for Mission-level or Project-level requirements, Level 2 for requirements at the system level, Level 3 for subsystem-level requirements, and Level 4 for internal, all-software, requirements.

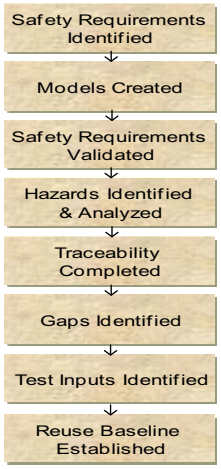

Figure 2 – IV&V Analysis Process

For this case study, safe hold was evaluated from the FM and FMEA artifacts. Through executing this process we discovered the safety requirements were potentially incomplete and there was room for improvement in FMEA and FM to eliminate gaps in the failure events and fault management. For example, the potential hazard caused by out-of-range errors in the Guidance, Navigation and Control calculation is not covered in the developer's FM or FMEA documents.

As part of the requirements validation effort, existing Levels 3 and 4 safety requirements were traced to FM and FMEA as well as the UML diagrams, and the failures in FM and FMEA were traced to each other. This traceability provides the foundation for the safety case at the highest level. See Figure 3.

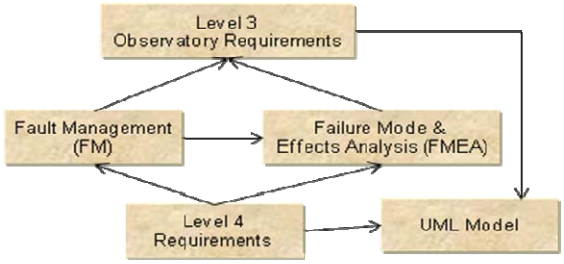

Figure 3 - High-Level Safety Case

This traceability helps determine if the hazards created by the failure event have the correct responses and if the responses satisfy all safety requirements in Levels 3 and 4 including the initiation of safe hold.

The gaps identified between the IV&V team's list of safety-critical events and those in the FM and FMEA serve as an indication of the developer's hazard identification sufficiency and hazard analysis adequacy, while the missing traceability provides an indication of the incompleteness of the software safety requirements. The identified gaps in both FM and FMEA as well as the missing traceability can be used to identify missing safety

requirements and are being documented as observations to be communicated to the developer of the satellite.

4 Conclusion

In this paper we present a proactive approach to the independent validation of safety requirements for systems of systems. Instead of checking to see whether a delivered system satisfies the intent of the derived safety requirements, the IV&V team first develops SRMs to capture their own understanding of the safety requirements and then proceeds to evaluate the project's hazard identification and hazard analysis effort for sufficiency and completeness of the safety requirements. We applied the proposed approach in support of a NASA science mission.

The case study illustrates, for the specific case of safe hold, how the NASA IV&V team assesses the safety requirements and safety documentation mandated by OSMA. This assessment provides the following:

- Builds a foundation for safety that is reusable
- Identifies room for improvement in the developer's FM documentation
- Identifies missing safe hold monitor initiating failure events
- Contributes goodness to any mission by improving mission safety and dependability

The study introduces a new way to independently validate software safety requirements, via the comparison of the FEMA and FM artifacts against the IV&V team's own list of safety-critical failures. Through this study we were able to identify potentially missing safe hold conditions and requirements in the project.

As the system, subsystem and components are traced in support of requirements validation, the requirements trace and the list of identified safety-critical failures provide essential information for safety-specific testing of the software.

The IV&V team's list contains a set of typical failure events that is reusable to identify and validate safe hold requirements in any space mission.

The safety case provides a portfolio of IV&V products that make up the foundation for reuse and contains the following artifacts:

- Failures leading to safe hold
- Safety requirements
- Models
- Activity diagrams for the fault monitoring and management and the enter safe hold use cases
- State transition diagram for the observatory state
- Missing safe hold and fault conditions
- Inputs to software testing
- SRM-generated test cases (work-in-progress)

Safe hold requirements are generic to all NASA space missions, many of which rely on a SoS comprised of legacy systems and new developments. While the safety case reported in this paper involves only one spacecraft, the artifacts from the safety case will provide a template for safety case development and model-based assessment for safe hold requirements that is applicable to any planetary orbiting SoS, with adjustment for custom payloads. We are developing a standardized process for the IV&V team to develop a portfolio of reusable artifacts for building safety cases for other safety-critical functions.

We plan to apply a similar approach to validate dependability attributes of the spacecraft software via the development of dependability cases.

5 References

[1] US Government Accountability Office, NASA: Assessments of Selected Large-Scale Projects, Report Number GAO-10-227SP, February 1, 2010. Accessed on 29 March 2010: http://www.gao.gov/products/GAO-10-227SP.

[2] Joint Software System Safety Committee, Software System Safety Handbook: A Technical and Managerial Team Approach, Joint Services Computer Resource Management Group, U.S. Navy, U.S. Army, U.S. Air Force, 1999.

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

[4] IEEE Std. 1012-2004, *IEEE Standard for Software Verification and Validation*, IEEE, 2004.

[5] D. Drusinsky, J.B. Michael and M. Shing, "A Framework for Computer-Aided Validation," *Innovations in Systems and Software Engineering*, 4(2), June 2008, pp. 161-168.

[6] T.P. Kelly, "A Systematic Approach to Safety Case Management," in *CAE Methods for Vehicle Crash Worthiness and Occupant Safety, and Safety Critical Systems*, SAE 2004 World Congress Special Publication SP-1870, Society of Automated Engineers, 2004.

[7] K. Cruickshank, J.B. Michael and M. Shing, "A Validation Metrics Framework for Safety-Critical Software-Intensive Systems," *Proc. 2009 IEEE International Conference on System of Systems Engineering*, Albuquerque, NM, 1-3 June 2009.

[8] A. Cockburn, *Writing Effective Use Cases,* Addison-Wesley, 2001.

[9] I. Alexander, "Misuse Cases: Use Cases with Hostile Intent," *IEEE Software*, 20(1), 2003, pp. 58-66.

[10] R. Weaver, The Safety of Software – Constructing and Assuring Arguments, Doctoral dissertation, University of York, Department of Computer Science, September 2003. Accessed on 30 July 2009: http://www.dtic.mil/doctrine/ jel/new_pubs/jp1_02.pdf.

[11] NASA IV&V Program, Portfolio Based Risk Assessment (PBRA) Process, Supporting Document S3106, Revision A, January 13, 2009.