Model Transformation for a System of Systems Dependability Safety Case

Judy Murphy

MPL Inc.
832 E. Woodland Tr.
Prairie du Sac, WI 53578
USA

jmurphy@mpl.com

Stephen B. Driskell

TASC Inc.
1000 Technology Dr.
Fairmont, WV 26554
USA

Stephen.Driskell@TASC.COM

Abstract *– Software plays an increasingly larger role in all aspects of NASA's science missions. This has been extended to the identification, management and control of faults which affect safety-critical functions and by default, the overall success of the mission. Traditionally, the analysis of fault identification, management and control are hardware based. Due to the increasing complexity of system, there has been a corresponding increase in the complexity in fault management software. The NASA Independent Validation & Verification (IV&V) program is creating processes and procedures to identify and incorporate safety-critical software requirements along with corresponding software faults so that potential hazards may be mitigated.*

This Specific to Generic … A Case for Reuse paper describes the phases of a dependability and safety study which identifies a new process to create a foundation for reusable assets. These assets support the identification and management of specific software faults and, their transformation from specific to generic software faults. This approach also has applications to other systems outside of the NASA environment.

This paper addresses how a mission specific dependability and safety case is being transformed to a generic dependability and safety case which can be reused for any type of space mission with an emphasis on software fault conditions.

Keywords: Reuse, Safety, Dependability, Validation, Verification, Model, Transformation

1 Introduction

The National Aeronautics and Space Administration (NASA) have a portfolio of major projects. These range from highly complex and sophisticated space transportation vehicles, to robotic probes, to earth orbiting satellites equipped with advanced sensors. In many cases, NASA's projects are expected to incorporate new and sophisticated technologies that must operate in harsh, distant environments. These projects have also produced ground-breaking research and advanced our understanding of the universe. However, one common theme binds most of the projects - they cost more and take longer to develop than planned [1].

Many of these systems function together as complex software intensive, safety-critical systems of systems (SoS) to support NASA's research missions in science, aeronautics, and human space flight exploration [2]. A SoS consists of multiple components and subsystems. A SoS development project is usually accomplished over a period of several years and most likely has rules, regulations and standards that must be followed. In NASA's case, it's imperative that all projects, including the SoS adhere to NASA's Office of Safety and Mission Assurance (OSMA) policies and procedures.

The mission of NASA's IV&V program, under the auspices of the OSMA, is to provide the highest achievable levels of mission and safety-critical software [9]. Safety assurance ensures that the requirements, design, implementation, verification and operating procedures for the identified software minimizes or eliminates the potential for hazardous conditions [3]. Software safety activities occur within the context of system safety, system development, and software development and assurance [3]. System safety assessment is a disciplined, systematic approach to the analysis of risks resulting from hazards that can affect humans, the environment, and mission assets [4]. The NASA IV&V Program provides assurance to our stakeholders and customers that NASA's mission-critical software will operate dependably and safely [2].

The IV&V program identified a need to address software-centric safety analysis and assess the quality of software safety engineering early in the development of a SoS to ensure the software manages safety requirements while not introducing system hazards. Most of the analysis is conducted via manual inspection, source code analysis, and more recently independent testing. The complex nature of software intensive systems introduces the challenge of narrowing the gap between the system requirements and their implementation. In spite of efforts to improve the flow of engineering information throughout the development process, oftentimes the implemented system does not fully match the required one, nor does it meet the user needs and

expectations [5]. This discontinuity between requirements documentation, software, and design implementation prevents sensible reusability, especially when the analysis is hardware specific.

The IV&V team develops its own independent understanding of each system under development which consists of a custom System Reference Model (SRM). These SRMs consist of a set of use cases, activity and sequence diagrams using the Unified Modeling Language (UML).

The basis of the modeling activity is derived from the review of artifacts provided by the SoS developer for each project. These artifacts include, but are not limited to, operations concepts, requirements, specifications, code, Failure Modes and Effects Analysis (FMEA), Fault Management (FM) and Failure Fault Analysis (FFA).

The IV&V analyses are model-based, striving to obtain goodness of product data in terms of three questions: *What is the system software supposed to do? What the system software is not supposed to do? What is the system software's expected response under adverse conditions?*

During Phase I of a multi-phase dependability and safety study, the SRM was extended to provide additional verification and validation. Specifically, for the spacecraft safe-hold (which is the autonomous software for managing spacecraft hazards without ground intervention) which establishes "safe" stable spacecraft operation, with minimal power consumption, power-positive (sun pointing), battery charging, and (for an earth orbiting mission) omni-directional (ground) communication. Ground operations can recover from the safe event by assessing the spacecraft failure and providing intervention to restore mission operations [2]. Safe-hold satisfies the cardinal rules for safety-critical software which are [6]:

- No single event or action shall be allowed to initiate a potentially hazardous event.
- When an unsafe condition or command is detected, the system shall:
    - Inhibit the potentially hazardous event sequence.
    - Initiate procedures or functions to bring the system to a predetermined "safe" state.

For the purpose of this paper, dependability and software safety are defined as:

Dependability:

- Dependability is the degree to which an item is capable of performing its required function at any randomly chosen time during its specified mission operating period, disregarding scheduled maintenance outages.

Software Safety:

- Software Safety is the aspect of software engineering and software assurance that provide a systematic approach to identifying, analyzing, and tracking software mitigation and control of hazards and hazardous functions (e.g., data and commands) to ensure safer software operation within a system [3].

The focus of the Phase I effort was a science satellite mission, with mature artifacts and a fairly comprehensive list of hardware fault conditions.

This paper addresses how a mission specific dependability and safety case is transformed to a generic dependability and safety case which can be reused for any type of space mission with an emphasis on software fault conditions.

The organization of the paper is as follows. Section 2 presents the process that was used for Phase I of the dependability and safety case. Section 3 describes Phase II of the dependability and safety case and transforms a SRM model from the specific to the generic. Section 4 describes how Section 3 can be applied to organizations both in and outside of the space program. Section 5 concludes with a discussion of future work. Section 6 includes the references.

2 Phase I: Overview of Initial Dependability & Safety Case

2.1 Model Safety-critical Behaviors

The NASA IV&V Program conducted a safety case study for a science satellite mission. Requirements validation was conducted on developer provided artifacts and a SRM was developed.

In addition, FMEA and FM artifacts were reviewed for fault conditions that would put the spacecraft in safe-hold. It was observed that the FMEA and FM artifacts largely focused on hardware failures. Those conditions were compared to an IV&V program developed list of safety-critical failure conditions which contained a combination of hardware and software faults. Traceability was created between system and software requirements and faults. From this activity, gaps were documented which helped identify missing safe-hold requirements.

Figure 1 portrays the IV&V analysis process created and followed. Figure 2 is a high-level depiction of the safety case which maps high-level safety requirements and lower-level safety requirements.

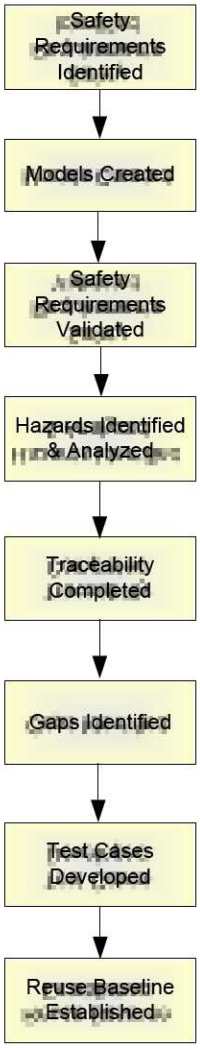


Figure 1 - IV&V Analysis Process [2]

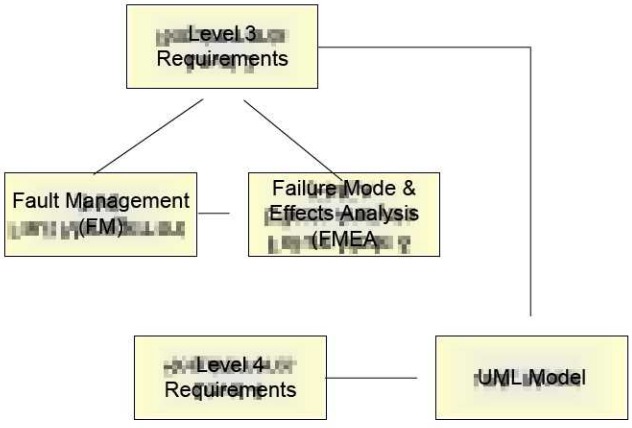


Figure 2 - High-Level Safety Case [2]

The identified gaps indicated that both the mission developer and the IV&V safety-critical failure conditions were incomplete. The IV&V list was updated and discussed with the mission developer who was very responsive to and is acting upon the data provided. Dependability and Safety were enhanced by the creation of a reusable template of artifacts and processes. These items make up the 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 [2].

Throughout this phase it became clear the process identified in Figure 1 was applicable to all of fault management and not just safe-hold and will be incorporated into future work addressed in Phase II.

3 Phase II: From the Specific to the Generic

The initial models and independent list of fault conditions were specific to a single science satellite mission. As an exercise, the independent list of fault conditions was compared to the fault conditions for the Mars Science Laboratory (MSL). Early on, it became obvious that the first mission Phase I IV&V list was only minimally re-usable due to its mission device-specific nature.

This is due to the fact that fault conditions for the Phase I science mission and the IV&V developed fault conditions list were device dependent. Although families of spacecraft may use the same underlying architecture, the subsystem device names are often different. It became clear that the models and the independent list of fault conditions needed to be based on the functionality of a subsystem at the highest level as opposed to the functionality of the device-specific hardware.

When comparing space missions to each other, it was immediately obvious that all missions share many of the same characteristics - regardless of the mission's purpose. All space missions have subsystems that deal with telemetry, command and data handling, guidance navigation and control, 1553 bus, temperatures, voltages etc. Functionality of other missions uses pyrotechnics, robotic rovers and unique experiments. Those subsystems may have differing designs and device names, but the subsystem functionality is the common thread.

Instead of focusing on the specific subsystem device with a specific fault, the focus will be on the functionality of a specific subsystem (Figure 3) with the fault conditions captured at a high and generic level to more easily be reused across other future missions. The generic behavior faults and related hazard management can be detailed later as the knowledge of the subsystem and its needs are discovered. This approach creates a standardized naming convention for dependability and safety cases across multiple system architectures that are reused.

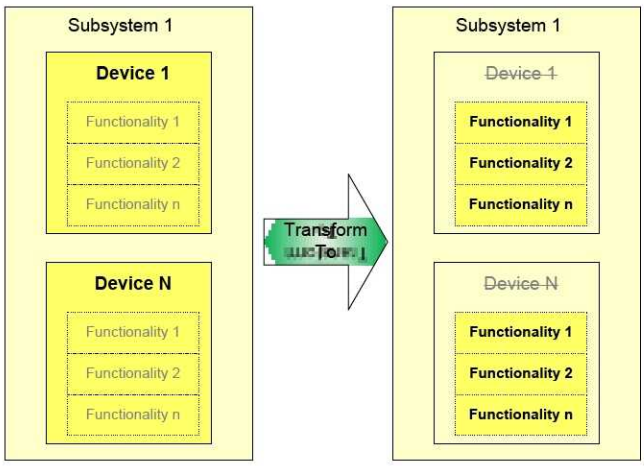


Figure 3 - Focus on functionality

Using a Command and Data Handling (C&DH) example from two distinctly different science missions, we determined the specific device name - Single Board Computer (SBC) from one mission and the specific device name - Flight Computer (FC) from another mission was actually the same device, the RAD750 computers (Figure 4) which we refer to as the main spaceflight computers.

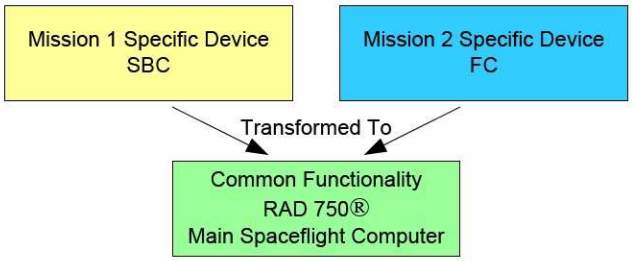


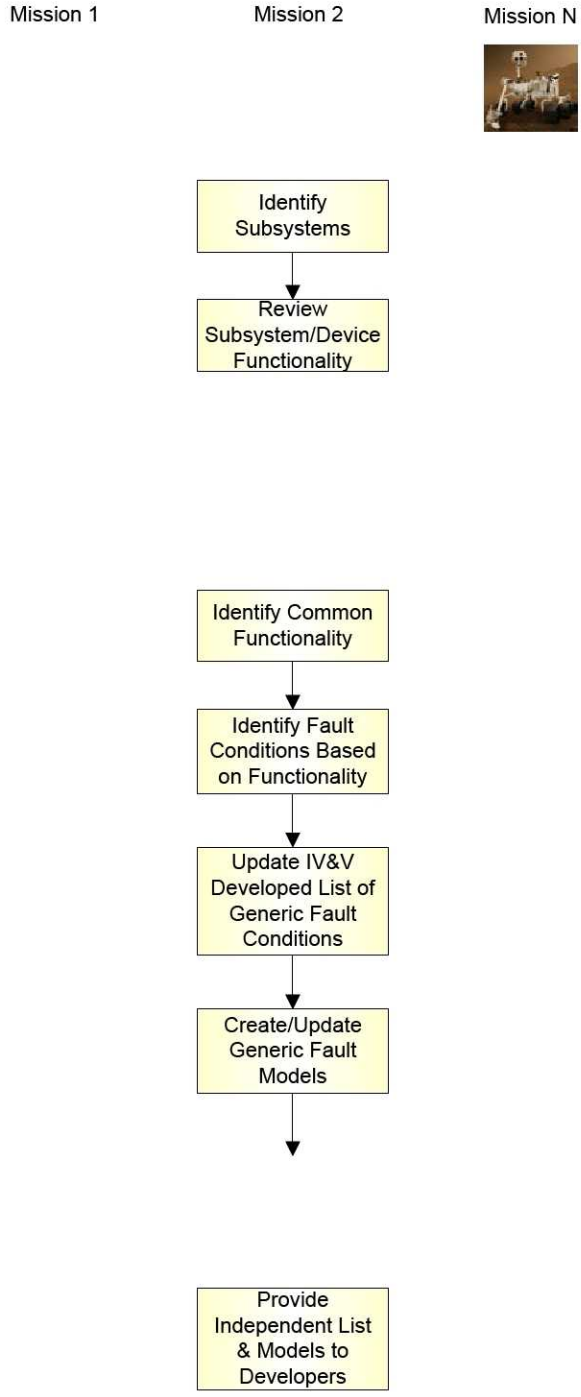
Figure 4 – Identifying common functionality

The RAD750® is commonly used for spaceflight and its functionality is well understood. This allows the re-use of fault conditions from one mission to another regardless of spacecraft family. This doesn't preclude or eliminate the development of, or the need for, additional fault conditions that are unique to a specific mission.

Moving forward, the fault conditions will be divided into three categories:

1. Cruise/orbit
2. Science Experiments
3. Surface operations (interplanetary rovers/landers)

The process in Figure 5 will be used to identify and communicate generic fault condition candidates.



Example

Figure 6 is an example of an activity diagram which depicts a high-level overview of fault management for a safe-hold event for a specific science mission. Each subsystem is comprised of specific devices in which specific failure(s) would result in a safe-hold event. Due to the proprietary nature of the data, the specific device names have been removed for the purposes of this paper.

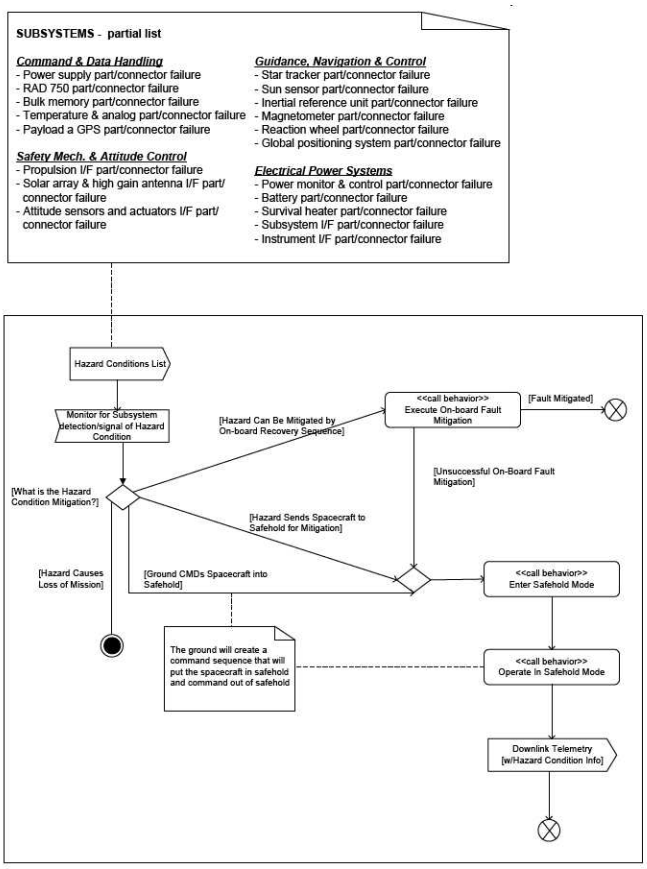


Figure 6 - Mission specific activity diagram for safe-hold fault management [2]

Figure 7 transforms Figure 6 into a generic model of fault management that can be applied to any space mission. Device names were replaced with the functionality of each subsystem which also account for software as well as hardware issues. The activities were modified to include faults of any kind, and are generic enough to be applied to and modified by any mission developer.

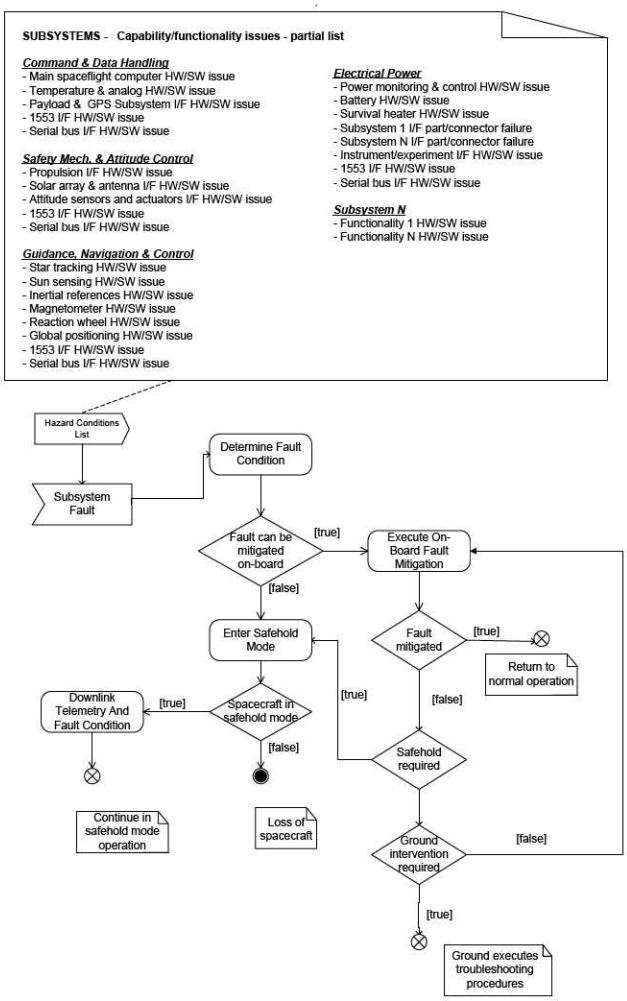


Figure 7 - Activity diagram for generic fault management

Once the subsystem functionality is understood, each unique function is identified and documented. The function is then decomposed into its known and potential hardware and software faults. Using the main computer processor as an example, potential faults include the following:

- Peripheral Component Interconnect (PCI) status register errors
- Excessive accumulation of uncorrectable SDRAM memory errors
- Overcurrent/undercurrent
- Overvoltage/undervoltage
- CPU halt/hung
- Etc

Some of these faults can be further decomposed into more detailed faults. The PCI status register is used to record status information for PCI bus related events [7] and bit 15 can be used to identify a parity error. Instead of referring to the "SBC" or the "FC," both of which are mission specific devices for two different science missions, we propose a reference to the "main spaceflight computer" along with potential faults that the mission developer can then apply to

their specific mission. This allows the mission developer to focus on the functionality and faults of a subsystem as opposed to being hindered by device names.

The benefits of this concept are multi-fold:

1. A pre-defined list of fault conditions provides a starting point for fault identification for any mission – doesn't recreate the wheel
2. A pre-defined list of fault conditions can be compared to existing fault conditions to identify gaps in both source requirements and elucidated faults – are the bases covered?
3. A pre-defined list of fault conditions increases the mission success and spacecraft safety probabilities through comprehensive contingency hazard management – is the list a superset of conditions? Is the hazard management adequate?

4 Applying Phase II to Your Project

Modifying Figure 5, your organization can apply similar fault management techniques even if your projects do not have the SoS complexity (Figure 8).

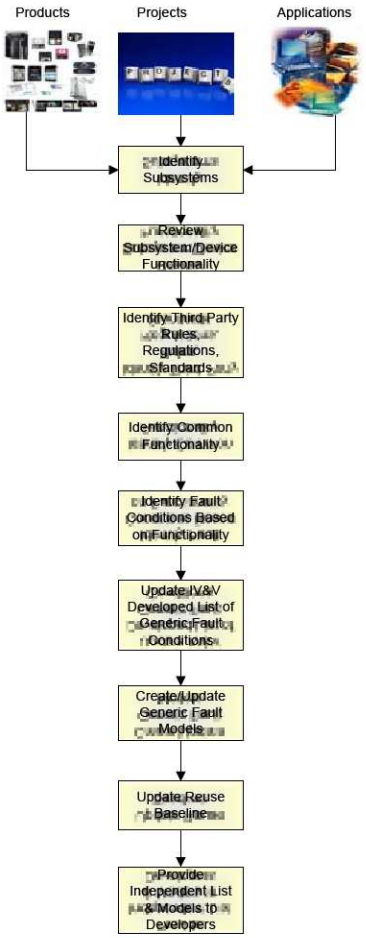


Figure 8 - Applying Phase II to your project

Replace the space mission examples with your system information. Decompose the system into subsystems (Figure 3) with a focus on subsystem functionality. Instead of reviewing device functionality, replace that with your own projects, programs or applications. When you identify common functionality, add your lessons learned from previous projects. These products are specific to your business and establish a reusable baseline of artifacts for use on multiple projects, where they become a library of analysis models.

Don't think spaceflight – think your business (Figure 9):

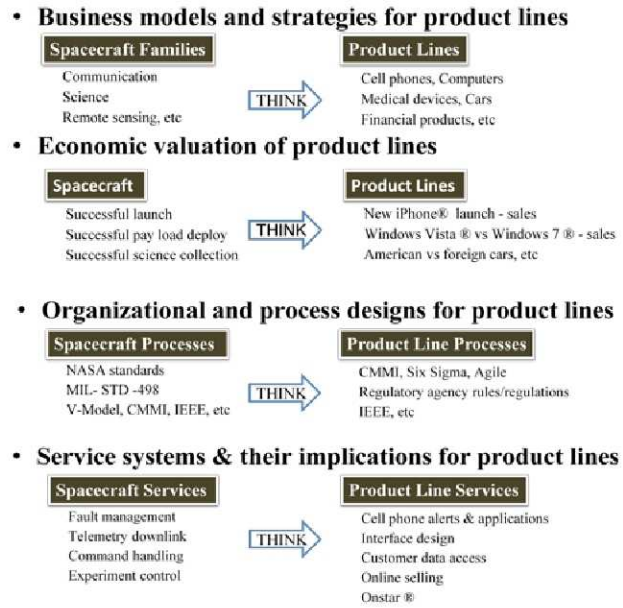


Figure 9 - Thinking about the same thing in a different way

This will allow your organization to efficiently identify fault conditions which accelerates your system and software development. This IV&V process approach ensures software and system quality. Using the IV&V three questions ensures that you have considered the special conditions necessary to verify that the software is properly implemented.

5 Conclusion

In this paper we present a proactive approach to identifying reusable fault conditions based on the functionality of a SoS instead of identifying fault conditions based solely on specific architecture implementation.

Phase II introduces a new way to independently validate software safety requirements, via the comparison of the FMEA, FM and FFA artifacts against the IV&V team's own list of fault conditions. This helps the mission developer ensure they have identified the correct fault conditions and will help the IV&V program keep their independent list current. This will also help the mission

developer identify missing requirements (shown as gaps by IV&V, through this developer interaction process). This will allow the IV&V program and the mission developer to do the following:

- Build a foundation for dependability and safety that is reusable
- Identify room for improvement in the IV&V program's processes that extends to all projects
- Identify missing fault condition initiating failure events, which result in hazards
- Identify missing safety requirements
- Contribute to the goodness of any mission

FM requirements are necessary for all NASA space missions. All of the mission's are a SoS comprised of a combination of legacy systems and new development [2]. Phase II has started with the creation of generic fault conditions for cruise/orbit portions of a mission.

Phase III will continue with fault conditions for experiments and Phase IV will continue with fault conditions for surface operations for planetary robotic missions.

Using the concepts provided in this paper, any organization can generalize for reuse their projects hazard controls using the process identified in Figure 3.

A future goal of this study is to build not only generic fault management monitors and controls by technology type, but to also provide automated models with technology to test, using Application Program Interfaces (APIs). This enables automation of the design and implementation validation and verification process. These modifications to manual analysis contribute to dependability, safety, availability, reliability performance, maintainability and security. It accelerates the attainment of these goals and other safety objectives. Accomplishing the approach outlined in this paper contributes to the reusability of software in general, as well as the IV&V processes used to ensure mission success.

6 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] S. Driskell, J. Murphy, J.B. Michael and M. Shing. "Independent Validation of Software Safety Requirements for Systems of Systems," *Proc. 2010 IEEE International Conference on System of Systems Engineering*, Loughborough University, UK, 22-24 June 2010.

[3] NASA Software Safety Standard (w/Change 1 dated 7/08/04), NASA-STD-8719.13B, March 12, 2008. Accessed on 14, November 2010: http://www.hq.nasa.gov/office/codeq/doctree/871913.htm.

[4] NASA General Safety Program Requirements (w/Change dated 7/20/09), NPR 8715.3C, March 12, 2008. Accessed on 14, November 2010: http://nodis3.gsfc.nasa.gov/npg_img/N_PR_8715_003C_/N_PR_8715_003C_.pdf.

[5] W. Gibbs, "Software's Chronic Crisis," *Scientific American*, pp. 86, Sep. 1994.

[6] NASA Software Safety Guidebook (w/Change 1 dated 7/08/04), NASA-GB-8719.13, March 31, 2004. Accessed on 14, November 2010: http://www.hq.nasa.gov/office/codeq/doctree/871913.htm

[7] PCI Local Bus specification, Revision 2.2, December 18, 1998. Accessed on 19, November 2010: http://www.ece.mtu.edu/faculty/btdavis/courses/mtu_ee3173_f04/papers/PCI_2d2.pdf.

[8] "NASA IV&V Vision and Mission." NASA IV&V Facility, accessed 9, December 2010, http://www.nasa.gov/centers/ivv/about/visionmission.html.