The Role and Quality of Software Safety in the NASA Constellation Program

Lucas Layman
Fraunhofer CESE
College Park, MD
e-mail: llayman@fc-md.umd.edu

Victor R. Basili, Marvin V. Zelkowitz
U. of Maryland and Fraunhofer CESE
College Park, MD
e-mail: {basili,mvz}@cs.umd.edu

***Abstract*— In this study, we examine software safety risk in the early design phase of the NASA Constellation spaceflight program. Obtaining an accurate, program-wide picture of software safety risk is difficult across multiple, independently-developing systems. We leverage one source of safety information, hazard analysis, to provide NASA quality assurance managers with information regarding the ongoing state of software safety across the program. The goal of this research is two-fold: 1) to quantify the relative importance of software with respect to system safety; and 2) to quantify the level of risk presented by software in the hazard analysis.**

We examined 154 hazard reports created during the preliminary design phase of three major flight hardware systems within the Constellation program. To quantify the importance of software, we collected metrics based on the number of software-related causes and controls of hazardous conditions. To quantify the level of risk presented by software, we created a metric scheme to measure the specificity of these software causes.

We found that from 49-70% of hazardous conditions in the three systems could be caused by software or software was involved in the prevention of the hazardous condition. We also found that 12-17% of the 2013 hazard causes involved software, and that 23-29% of all causes had a software control. Furthermore, 10-12% of all controls were software-based. There is potential for inaccuracy in these counts, however, as software causes are not consistently scoped, and the presence of software in a cause or control is not always clear. The application of our software specificity metrics also identified risks in the hazard reporting process. In particular, we found a number of traceability risks in the hazard reports may impede verification of software and system safety.

Keywords-Constellation program; hazard reports; measurement; safety; empirical software engineering

I. INTRODUCTION

Development of large complex systems in the aerospace, defense, energy and travel industries requires constant attention to safety risks. A safety risk is a risk whose effect can be injury or the loss of life either directly or through a chain of events. The task of controlling safety risk is further complicated when developing complex systems due to the amount of planning, assessment and communication required to manage multiple projects. The issue of controlling software safety risk has become a leading area of concern in systems development as many traditionally hardware-centric systems become more reliant on software.

In this paper, we examine issues of measuring software safety in one such system – the multi-year, multi-billion dollar Constellation spaceflight program at NASA. Software plays a significant role in many Constellation systems. Analysis, planning, and mitigation of safety risks pervade every phase of development. The multitude of systems that comprise the Constellation program are developed by contracting organizations using a form of concurrent engineering [5] wherein multiple development activities (i.e. design, implementation, testing) occur in parallel. For NASA quality assurance managers, obtaining an accurate, program-wide picture of software safety risk is difficult across these multiple, independently-developing systems for a number of reasons:

- There are many development groups, each with their own reporting style for safety risks. Even though program-wide standards exist, each group has their own interpretation of how to address those standards.
- The NASA panel charged with overseeing system safety has limited resources and technical knowledge to fully understand all the implications of software safety. Although these experts have significant experience managing the mostly non-software development of rockets and spacecraft at NASA, applying the NASA safety process to software is relatively new.
- The safety engineers responsible for the systems sometimes have limited understanding of how to describe software safety risk to meet the requirements of NASA safety reviews.
- The rules for recording software risk in the safety tracking systems were only recently developed, resulting in no clear delineation between software-based risks and non-software-based risks.

NASA Safety Reliability and Quality Assurance personnel (SR&QA) have undertaken a number of initiatives to address these challenges. As part of these initiatives, we have participated in the development of a software safety measurement program to provide a program-wide overview of software safety risk. The goal of our study was to leverage one source of safety information, hazard analysis, to provide NASA quality assurance managers with information on the ongoing state of software safety. We collected data to demonstrate the increased role of software as a control mechanism for safety risks, suggesting that increased emphasis on software safety analysis is warranted. The software safety metrics also yielded new guidelines for

safety engineers to use when describing software safety risks.

The remainder of this paper is organized as follows. In Section II we describe the research context and provide an overview of the hazard analysis process on the Constellation program. In Section III we describe the measures we capture on software risk described in hazard reports. In Section IV we present the data findings of this study and in Section V we discuss these results and a several risks uncovered in the software safety reporting process. Finally, in Section VI we give our conclusions on how such measures can best be used in this environment.

II. Research Context

The Constellation program is a complex *system of systems* (see Figure 1 for the Constellation program hierarchy). Each *system* contains multiple *elements*, each with numerous, complex hardware and software *subsystems*. Our research focuses on the hazard analysis of three projects (A, B, C), one at the system level and two at the element level. The names of the projects are kept anonymous for confidentiality purposes. The scale and complexity of the Constellation program presents many challenges to NASA personnel in assessing the state of system safety and, in particular, software safety.

A. The safety review process in the Constellation program

Systems development is overseen by NASA and follows a government acquisition V-model where individual system development is performed by multiple contract companies. The multiple Constellation systems are being developed in parallel over a period of several years. The development groups use various databases for coordinating information among the groups. These include databases containing project requirements, safety hazards, defects, designs and other project management information.

At various times, checkpoint meetings are held by the Constellation Safety & Engineering Review Panel (CSERP), which acts as gatekeeper for development milestones. There are several milestones in the development process (e.g. system requirements review, preliminary design review, critical design review) with different requirements for the type of system and software safety analysis that must be performed. At each milestone, the development groups identify safety risks in system operation and design and create strategies (controls) for mitigating those risks. The CSERP reviews the risks and the operational or design strategies for mitigating these risks. The CSERP panel then approves the current design or requests changes to provide for better risk mitigation. As development progresses and the system matures, the analyses and designs become more specific and concrete. The primary responsibility of the CSERP is to ensure that all safety risks which could result in loss of life, loss of the vehicle, or loss of mission are identified and handled properly.

Analyzing and designing to mitigate software risk is supported by SR&QA personnel. SR&QA is a division within the Constellation program that provides guidance to safety engineers on the specific projects and participates in CSERP safety reviews. This division is comprised of NASA employees and contractors with expertise in hardware, software and mission assurance.

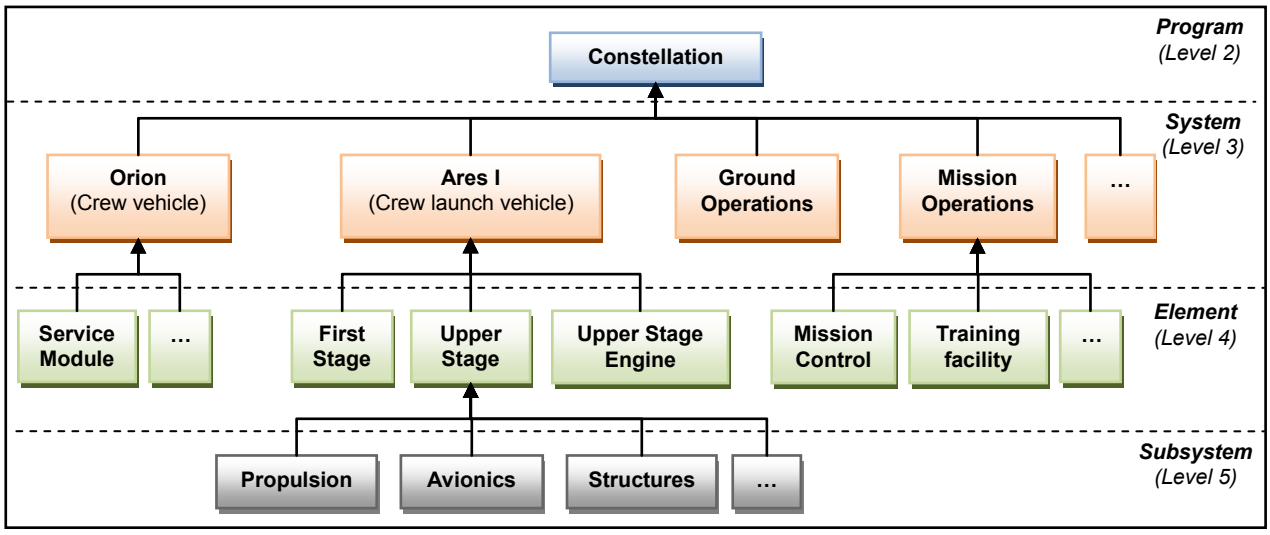

Figure 1. Constellation program hierarchy (abbreviated example)

B. Hazard analysis in the Constellation program

Our study thus far focused on the *hazard analysis* methodology used in the program. Hazard analysis is a top-down approach to system safety analysis. In the Constellation program, hazard analysis is complemented by other safety analysis methods, including failure modes and effects analysis (FMEA), fault-tree analysis (FTA), probabilistic risk assessment (PRA), quality audits, process and project metrics, and many more.

A *hazard* is any real or potential condition that can cause: injury, illness, or death to personnel; damage to or loss of a system, equipment, or property; or damage to the environment. An example of a hazard might be "Avionics hardware failure leads to loss of mission." The hazard is accompanied by a list of systems, elements and subsystems that cause or are affected by the hazard, a detailed description of the hazardous condition, and information regarding the likelihood of the hazardous condition occurring.

Hazards are described with several important properties:

- *Causes* – The root or symptomatic reason for the occurrence of a hazardous condition;
- *Controls* – An attribute of the design or operational constraint of the hardware/software that prevents a hazard or reduces the residual risk to an acceptable level;
- *Verifications* – A method for assuring that the hazard control has been implemented and is adequate through test, analysis, inspection, simulation or demonstration.

Figure 2 illustrates the conceptual organization of a hazard. Each hazard (e.g., engine failure) has one or more causes (e.g., failure with fuel line, software turns off the engine, or physical failure of engine). Each cause has one or more controls that reduces the likelihood that a cause will occur or mitigates the impact should the cause be realized (e.g., backup computers to account for software failures). Each control has one or more verifications (e.g. test, inspection, simulation or demonstration) to ensure that the control is appropriately implemented. In the Constellation program, all hazards and their associated causes, controls and verifications are stored in a database called the *Hazard Tracking System* (HTS). Each such hazard is stored as a *Hazard Report* (HR) in the HTS.

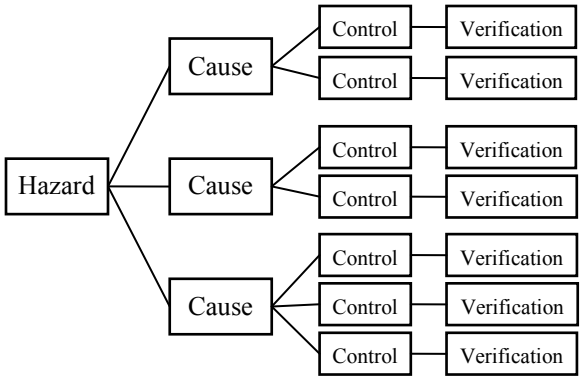

Figure 2. Hazard structure

Controls often represent new requirements for the system. For example, if a buffer overflow causes a software component to overflow (e.g., the inertial guidance computer overflow in the Ariane 5 rocket failure in 1996 [1]), a possible control could be to monitor the values of the appropriate register and provide some mitigating action if it overflowed. This would represent a new requirement that the developers would have to implement in the flight software.

Causes and controls can also be *transferred* to another cause or control in a different hazard reports. Transfers imply that the cause or control is fully described in the other hazard report. For example, a structural collapse will impact nearly every system in a hazard. Rather than list causes and controls for structural collapse in every hazard report, it is handled in its own report that is referred to by the other hazards. During system implementation, all transferred causes and controls must be verified for a hazard reports to be considered "closed." Verifying transfers is a manual, labor intensive process and is at risk when transfer references are not kept up to date.

It is important to note that, in this environment, software is never a hazard; hazards all represent physical events that may harm the mission. Component failure (e.g., degraded thruster performance) or outside events (e.g., hitting space debris, impact of weather, cosmic ray impact) may impact a mission, but software itself is not a hazard. However, software, as well as human error or component failure, can certainly *cause* a hazard (e.g., the software shutting a fuel valve at the incorrect time). Therefore, the HTS describes physical events, some of which are caused by software and some of which are not.

We define a *software hazard* as a hazard that contains one or more software causes. A *software-related hazard* is a hazard where software is either one of the causes or software is in one or more of the controls. We are interested in software-related hazards because, even though software may not be a direct cause of a hazard, software that is part of the control can be faulty and cause a subsequent hazard (e.g., Therac-25 radiation errors [4]). Software hazards are a proper subset of the software-related hazards. Both software hazards and software-related hazards may include hardware causes and controls as well.

III. Research Method

We examine the Constellation hazard reports to assess the state of software safety risk and provide feedback into the software safety process. Our approach [3] is based on the premise that there is a relationship between the processes used during software development and the product's characteristics. Not achieving the anticipated product characteristics (e.g. safe software) may be the result of not adhering to the process, or the process itself may be flawed. By analyzing the execution of the software safety process in the Constellation program, we hope to gain insight into whether appropriate software safety processes are being performed and performed appropriately.

Process artifacts available during development can provide insight into process execution. Hazard reports, produced throughout system development, are one artifact that can provide insight into the state of software safety and can be used to evaluate the software safety process. We describe our goals for evaluating software safety risk using the Goals Questions Metrics (GQM) model [2]:

1. Analyze a sample of the hazards reported for Projects A, B and C in order to characterize them with respect to the prevalence of software in *hazards*, *causes*, and *controls* from the point of view of NASA quality assurance personnel in the context of the Constellation program.
2. Analyze the *software causes* in a sample set of hazard reports for Projects A, B and C in order to evaluate them with respect to the specificity of those software causes and hazards from the point of view of NASA quality assurance personnel in the context of the Constellation program.

Specificity in hazard causes is important for understanding concrete, verifiable controls. A lack of specificity in the definition of causes indicates a risk that the cause has not been adequately identified and evaluated so as to be controlled. Each system we analyzed had 1-2 *"generic" software hazard reports*, which describe only the procedures for how software should be developed, but do not describe specific design or behavior. Often, software causes had a single control referring only to these "generic" reports rather than a specific design attribute. These controls represent risk in that there is no objective verification that a software cause has been controlled by adhering to the software process.

An important note: our evaluation focuses on software *safety* risk. Safety, in the Constellation program, does not include software *security*. Software security on the Constellation program is handled a by separate organization charged with hardening the software systems against malicious attack and assisting in secure software design.

A. Goal 1: Quantify the prevalence of software in hazards, causes and controls

Goal 1 is useful to SR&QA personnel in understanding the risk analysis effort required to adequately control software risk and also to identify systems and subsystems that involved more software risk than others. We first analyzed the hazard reports to classify causes of a hazard and controls for these causes as either *software* or *non-software*. These data can then be used to answer a number of questions (see below). Tables I and II show the categories of causes.

1. What percentage of the hazard causes are software causes? This is represented by entries B and C of Table I. The percentage is given by (B+C)/(A+B+C+D).
2. What percentage of the hazards is software–related? Does software play a role in either causing or mitigating a hazard? Those hazards are represented by the entries in boxes X, Y and Z of Table II. The percentage is given by (X+Y+Z)/(W+X+Y+Z).
3. What percentage of hazard causes are non-software (e.g., hardware, operational error, procedural error) causes with software controls, i.e., Z in Table II? These are potentially "hidden" software risks. For example, if software, as a control, is monitoring a condition, if the monitoring software fails, even though the hardware is functioning correctly, there is a risk that the monitor will fail to detect an actual subsequent problem or the software may send erroneous status messages. Thus, the software can again be the cause of a hazardous condition. The percentage is given by D/(A+B+C+D).
4. What percentage of non-software causes contains software controls? D/(A+D)
5. What percentage of the causes contains software controls? (C+D)/(A+B+C+D)
6. What percentage of causes is transferred, i.e., another HR contains the control for this cause? Transferred causes can incur risk when traceability is not maintained.
7. What percentage of controls is transferred? Transferred controls also introduce traceability risks.
8. What percentage of the non-transferred hazard controls is specific software controls, i.e. describe software behavior or design? This may be a simple measure of the number of software related requirements that deal with safety critical software.
9. What percentage of non-transferred controls are references to "generic" software controls?

TABLE I. CAUSE CATEGORIES

Cause Table	**Causes**		
Controls		**Non-software**	**Software**
	Non-software	(**A**) Non-software causes with no software controls	(**B**) Software causes with no software controls.
	Software	(**D**) Non-software causes with at least one software control	(**C**) Software causes with at least one software control

TABLE II. HAZARD CATEGORIES

Hazard Table	**Causes**		
Controls		**Non-software**	**Software**
	Non-software	(**W**) Hazards with no software causes or controls	(**X**) Hazards with at least one software cause and no software controls
	Software	(**Z**) Hazards with no software causes and at least one software control	(**Y**) Hazards with at least one software cause and one software control

B. Goal 2: Evaluate the specificity of software causes

Goal 2 assists SR&QA personnel by identifying software hazards and software causes that require additional work on the part of the safety engineers. Furthermore, hazard reports mature over time, and the evaluation of Goal 2 enables SR&QA personnel to track the maturation of software causes as the systems approach their quality milestones. Goal 2 is in some respects a proxy for SR&QA and CSERP personnel, who must understand the origin of a hazardous condition (the cause) as described in the hazard reports. Software causes are evaluated according to their *specificity*.

As nearly all projects are in preliminary design, we focus on evaluating *causes* since only causes, which should be well-defined for a Phase I safety review prior to Preliminary Design Review (PDR). Based on the preliminary analysis of software-related hazards, we have derived an initial set of software safety metrics that can be applied to hazard reports. These metrics were developed using feedback from SR&QA personnel and using existing hazard reports as examples.

By PDR, the software is defined to the level of Computer Software Configuration Item (CSCI). We judge the minimal specificity of software causes based upon the existence of three attributes in the cause description: (1) which software component may fail its intended operation (*origin*), (2) what is the erroneous behavior for this software component (*erratum*), and (3) what impact does this erroneous behavior have on the system (*impact*)? We then define a metric to evaluate the specificity of a cause.

CSCIs (e.g. Guidance Navigation & Control, Vehicle Management) are more specific than sub-systems (e.g. avionics, propulsion) and would represent Level 6 and below in Figure 1, enabling the analysis of more specific causes and corresponding controls. Furthermore, CSCIs can be used in the analysis of relationships between components and specifying the safety-critical events, commands and data. Describing software causes at the CSCI level enables the hazard analyst to identify specific design elements that satisfy the requirement for controls.

Software-related causes and sub-causes may be described in a single cause in a hazard report or they may appear as multiple separate causes. From the software cause and sub-cause metrics for a given hazard, an overall hazard rating for each hazard report for software causes can be created.

1. For each hazard report, what are the number and percentages of L1, L2 and L3 causes where L1, L2 and L3 are defined as:
   - **L1**: a specific software cause or sub-cause for a hazard, where a specific software cause *must* include all of the following:
     - Origin – the CSCI that fails to perform its operation correctly
     - Erratum – a description of the erroneous command, command sequence or failed operation of the CSCI
     - Impact – the effect of the erratum where failure to control results in the hazardous condition, and if known, the specific CSCI(s) or hardware subsystem(s) affected
   - **L2**: a partially-specified software cause or sub-cause for a hazard, where a partially-specified software cause specifies one or two of the origin, erratum or receiver at the CSCI/hardware subsystem level.
   - **L3**: a generically defined software cause or sub-cause for a hazard, where a generically-defined software cause does not specify the origin, erratum or receiver at the CSCI/hardware subsystem level.
2. For each system, what are the number and percentages of La, Lb, Lc, Ld and Le hazards where La-Le are defined as:
   - **La**: All software causes and sub-causes in a hazard are L1
   - **Lb:** all software causes and sub-causes in a hazard are L1 except for a single L3
   - **Lc**: Software causes and sub-causes are a mix of L1, L2 and L3 with at least one L1
   - **Ld**: All software causes and sub-causes are either L2 or L3 with at least one L2
   - **Le**: All software causes are L3

A low hazard rating (e.g., Ld and Le) may indicate there is a risk of not being able to mitigate the software risk associated with these hazards. A high rating (e.g., La and Lb) more likely indicates that the development team fully understands the risk and has addressed it appropriately. The overall hazard ratings provide a top level view of the maturity of software cause specificity in a subsystem or mission element.

We note that *these ratings do not measure the quality or the completeness of the software cause and control analysis*; these ratings only reflect the specificity of the information captured in the hazard reports. We believe that these ratings likely indicate risk where insufficient specificity has been provided to identify the software-based cause of a hazardous condition within the hazard report. Insufficient specificity probably indicates that the problem is not well understood, unless further details are included in supporting documentation. However, unless such supporting information (and the necessary context and expertise to interpret it) are maintained with the hazard report, there is risk that information will be lost.

C. Data sources and context

A total of 154 hazard reports were analyzed for three Constellation systems: 77 in the Project A, 57 in the Project B, and 20 in Project C. Project A is developed by NASA while Projects B and C are being developed by contractor organizations. The three projects are developing large, flight hardware systems. Software is a critical element in controlling the function of these systems, and the amount of software varies significantly in each project.

Each project performed their hazard analyses according to Constellation program guidelines. During the Preliminary Design Phase, the safety engineers for each project met with the CSERP to evaluate, discuss, and mature the hazard reports. Additional description of the three elements is presented with the study findings below. Our analysis includes all available hazard reports available from the Preliminary Design Phase of these systems with the exception of four hazard reports from Project A that were in an incomplete, preliminary state.

D. Analysis procedure

We first identified software and non-software causes and controls in the hazards reports. An analysis of the text of each hazard report was performed manually as follows:

1) Each cause from a hazard report was entered in a separate row of an Excel spreadsheet (see Table III).

2) For each cause, each relevant control description was marked as either a software control (green), a non-software control (blue), a control that transferred to another hazard report (orange), a transfer to another hazard report detailing exclusively with software (yellow), or controls with multiple sub-controls (grey). A control was determined to be a software control if it described the behavior or design of FCSW (flight computer software), FC (flight computer), specific CSCIs, or used the word "software."

   a) Each control in the hazard report corresponds to a column in the spreadsheet.

   b) Some software controls contained enumerated "sub-controls" that described separate software design and behavior. The parent control was marked as grey, and the sub-controls were separately listed in the spreadsheet in the order of appearance in the hazard report. These sub-controls were classified using the same scheme as in step 2.

3) The cause description for each cause was read and was marked as either a software cause (green) or a non-software cause (white). Causes for which *all* controls were transferred were marked red and excluded from further analysis under the assumption that the cause was controlled by the transferred hazard report(s). A cause was determined to be a software cause when software, FCSW (flight computer software), FC (flight computer) or specific CSCIs were mentioned in the cause description. Table III provides an example of the cause-control matrix for a hazard report.

The classifications of causes and controls were then counted for each hazard report and recorded in a separate worksheet (see Table IV for an example). These data were then used to compute summary statistics across all hazard reports and to answer the questions posed in Section 2.A. The "causes" column is the total number of causes listed in the hazard report, and the "active causes" column is the number of non-red causes in the cause-control matrix. Transferred controls were counted only when other non-transferred controls existed for the same cause. For example, Cause 6 in Table III was counted as having two controls since the transferred control appears with a non-transferred control, whereas Cause 4 was counted as having no controls. When transferred generic software controls (yellows) were counted, they were counted as software controls (e.g. Cause 2 in Table III was counted as having three software controls).

TABLE III. CAUSE-CONTROL MATRIX EXAMPLE

Hazard Report	Cause	Controls																
		1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*
HR1	1																	
	2																	
	3																	
	4																	
	5																	
	6																	
	7																	
	8																	
	9																	

SW | Non-SW | transferred | Transferred to generic SW HR

TABLE IV. EXAMPLE TABULATION OF CAUSES AND CONTROLS

Hazard Report	Causes	Active causes	Software causes	Non-SW causes with SW controls	Controls	SW controls	HR is SW related?
HR1	9	5	4	1	12	4	TRUE
HR2	14	10	0	6	33	7	TRUE

IV. FINDINGS

These data were calculated for all of the hazard reports available for each element. The team reviewed a total of 154 hazard reports, 2013 causes, and 4916 controls. Tables V-VII are the results for Project A, Tables VIII-X are the results for Project B and Tables XI-XII are the results for Project C.

TABLE V. PROJECT A HAZARD TABLE

	Non-software cause			At least 1 software cause		
no software control	**W**	39	*51%*	**X**	0	*0*
at least 1 software control	**Z**	10	*13%*	**Y**	28	*36%*
Total	77					

TABLE VI. PROJECT A CAUSE TABLE

	Non-software cause			Software cause		
no software control	**A**	393	*71%*	**B**	0	*0%*
at least 1 software control	**D**	76	*14%*	**C**	85	*15%*
Transferred causes	252					
Total	806					

TABLE VII. PROJECT A CONTROL TABLE

	N	% of total	% of non-transferred
Non-software	1603	64%	82%
Software	243	10%	12%
Generic software controls	105	4%	5%
Transferred controls	566	22%	-
Total	2517	100%	

TABLE VIII. PROJECT B HAZARD TABLE

	Non-software cause			At least 1 software cause		
no software control	**W**	19	*33%*	**X**	0	*0%*
at least 1 software control	**Z**	1	*2%*	**Y**	37	*65%*
Total	57					

TABLE IX. PROJECT B CAUSE TABLE

	Non-software cause			Software cause		
no software control	**A**	398	*77%*	**B**	0	*0%*
at least 1 software control	**D**	57	*11%*	**C**	62	*12%*
Transferred causes	155					
Total causes	672					

TABLE X. PROJECT B CONTROL TABLE

	N	% of total	% of non-transferred
Non-software	1799	75%	84%
Software	298	12%	14%
Generic software controls	37	2%	2%
Transferred controls	265	11%	-
Total	2399	100%	

TABLE XI. PROJECT C HAZARD TABLE

	Non-software cause			At least 1 software cause		
no software control	**W**	6	*30%*	**X**	0	*0%*
at least 1 software control	**Z**	0	*0%*	**Y**	14	*70%*
Total	20					

TABLE XII. PROJECT C CAUSE TABLE

	Non-software cause			Software cause		
no software control	**A**	275	*81%*	**B**	0	*0%*
at least 1 software control	**D**	9	*3%*	**C**	57	*17%*
Transferred causes	194					
Total	535					

A. *Metrics summary*

From these data, we calculate the metrics necessary to help answer the questions from Section III that help quantify the importance of software with respect to system safety (see Table XIII). These data demonstrate that although a small percentage (12-17%) of hazard causes are software causes, the percentage of hazardous conditions that are either caused by software or are controlled by software in much higher (49-70%). This indicates that software is a safety-critical aspect of the overall system and over half of all hazard reports are software-related.

Note that while 49% of Project A's hazard reports are software-related, 67% of Project B hazard reports and 70% of Project C hazards are software related. This disparity can be a consequence of the characteristics of the three systems, an indication of how the three development organizations arrange the subjects of the hazard reports differently, or a combination of these. The lack of a uniform structure for reporting software-related hazards inhibits a consistent, general methodology for software risk assessment based on the hazard reports. In all three systems, the importance of software clearly demonstrates the need for a strong software development process with adequate control and verification.

TABLE XIII. SUMMARY METRICS QUANTIFYING THE IMPORTANCE OF SOFTWARE WITH RESPECT TO SYSTEM SAFETY

	Question	Project A	Project B	Project C
1	What percentage of the hazard causes are software causes?	15%	12%	17%
2	What percentage of the hazards is software-related?	49%	67%	70%
3	What percentage of hazard causes are hardware causes with software controls?	14%	11%	-
4	What percentage of hardware causes has software controls?	16%	13%	-
5	What percentage of the causes has software controls?	29%	23%	-
6	What percentage of causes is transferred?	31%	23%	36%
7	What percentage of controls is transferred?	22%	11%	-
8	What percentage of the non-transferred hazard controls are specific software controls?	12%	14%	
9	What percentage of the non-transferred hazard controls are references to "generic" software controls?	5%	2%	-

B. *Software hazard and software cause specificity ratings*

To quantify the level of risk presented by software, we applied the software cause and software hazard metrics described in Section III to the causes in the hazard reports. The L-metrics are applied to software causes and hazards with at least one software cause only (Tables XIV-XVI).

Using the current software cause metric definitions, there are noticeable differences between elements. Project A had a greater proportion of well-specified software causes than Projects B and C at the time of analysis. Project B had a large portion of software-causes that could be considered "in work," and thus one would expect the distribution to shift to the higher end of the scale as work progresses. Project C, however, was non-specific in terms of software causes. In general, the software causes in Project C had very specific descriptions of the impact of a software failure on the non-software, but little was described in terms of what caused the software to malfunction or how the software error manifested beyond stating that "the software fails."

TABLE XIV. PROJECT A SOFTWARE SPECIFICITY RATINGS

Hazard ratings		
La	5	18%
Lb	7	25%
Lc	7	25%
Ld	3	11%
Le	6	21%

Cause ratings		
L1	65	50%
L2	26	20%
L3	38	29%

TABLE XV. PROJECT B SOFTWARE SPECIFICITY RATINGS

Hazard ratings		
La	3	8%
Lb	1	3%
Lc	14	38%
Ld	13	35%
Le	6	16%

Cause ratings		
L1	64	38%
L2	68	40%
L3	37	22%

TABLE XVI. PROJECT C SOFTWARE SPECIFICITY RATINGS

Hazard ratings		
La	0	0%
Lb	0	0%
Lc	0	0%
Ld	12	86%
Le	2	14%

Cause ratings		
L1	0	0%
L2	41	71%
L3	16	29%

Once again, we note that *these ratings do not measure the quality or the completeness of the software cause and control analysis*; these ratings only reflect the specificity of the information captured in the hazard reports. We believe that what these ratings reflect may indicate risk where insufficient specificity has been provided to identify the software-based cause of a hazardous condition within the hazard report. Insufficient specificity may indicate that the problem is not well understood.

V. RESULTS

In Section III we presented our two GQM goals for this study. We describe our conclusions for each goal in Section V. A. We describe and discuss a number of risks observed in the hazard reports and their implications for software safety in Section V.B.

A. *Conclusions of GQM evaluation*

GQM Goal 1: *Analyze a sample of the hazards reported for projects A, B and C in order to characterize them with respect to the prevalence of software in hazards, causes, and controls from the point of view of NASA quality assurance personnel in the context of the Constellation program.*

It is clear from Section IV.A and Table XIII that software plays a significant role in the safety of the Constellation program. However, there is variable precision in the counting of software hazards, causes and controls; the guidelines for reporting hazards are open to interpretation and each group reported and scoped hazards differently. Furthermore, the number of software and software-related hazards is likely greater than shown as there may have been software causes and controls that were not identified as such.

From the point of view of the quality assurance personnel, it is difficult to track each hazard cause and control to its source, and overall traceability becomes more difficult. In Section B below, we present our lessons learned about the nature of software in hazard reports, which directly addresses the issue of traceability and proposes some modifications to the development process to take this into account.

GQM Goal 2: *Analyze the software causes in a sample set of hazard reports for projects A, B and C in order to evaluate them with respect to the specificity of those software*

causes and hazards from the point of view of NASA quality assurance personnel in the context of the Constellation program.

Section IV.B and Tables XIV-XVI provide the summary data for this goal. For all 3 projects, the causes were rated at least Level L2 for 70-78% of all causes. However, project C had 0% with the highest rating of L1, whereas the other two projects had L1 ratings of 38% and 50%. Although this data is only from PDR, project C has significantly less specificity that the other two. Part of the CSERP review process is to further refine the specificity of hazard reports over time, particularly in subsequent development phases. These current figures provide an important baseline for CSERP and SR&QA personnel to monitor the progress of hazard report specificity over time.

Because this data was collected during the time for PDR, the process used by the Constellation Program did not require that controls and verifications be fully developed. For this reason, we were unable to fully characterize controls and their verifications for this study and it will have to wait for a later time.

B. *Informing the software safety process: risks observed in software hazard reports*

In the process of developing this data, we have uncovered a number of potential risks for the program with regard to how software-related hazards are reported. A lack of completeness and uniformity in the information describing software risk (i.e. software causes and controls) creates a risk in itself. That is, software causes may not be thoroughly described and understood, and controls that involve software may be difficult to verify.

Maintaining traceability between causes, controls and verifications is essential for ensuring that all causes of hazardous conditions have a control in place and that this control has been verified. Our analysis was conducted on hazard reports that had passed or were in the Phase I CSERP review where complete description of controls and verifications were not yet required. However, traceability needs to be maintained between causes, controls and verifications during Phase I. We observed a number of risks associated with incomplete traceability in the hazard reports.

1) ***Risk 1*** *– Lack of consistency in structuring hazard report content, causes and control descriptions impairs understanding.*

All hazard reports in the Constellation program follow a standard template, but the content of the hazard reports, cause descriptions, and control descriptions differed substantially between the three programs and between hazard report authors within the same program. In some cases, the unstructured text creates risk that the CSERP may not be able to fully understand the risks detailed in the hazard even with supporting materials.

During preliminary design, safety engineers are still developing first versions of hazard reports and becoming familiar with the expectations of CSERP and the requirements of the software safety process. This risk has abated over time as CSERP and SR&QA personnel have worked closely with safety engineers to form a uniform expectation for hazard report content. These experiences are also being used to recommend improvements to NASA process documentation and training materials.

2) ***Risk 2*** *– Lack of consistent scope in causes and controls impairs risk assessment.*

Related to Risk 1, there is a lack of uniformity in scoping *software* causes and controls between programs or between hazard reports within programs in some cases. A cause reading "Generic avionics failure or software flaw causes improper operation of control thruster" certainly involves software, but it is not scoped to a particular software component as required by NASA procedure.

Much of this risk can be attributed to unfamiliarity with describing software risk in hazards and misunderstanding the expectations of the CSERP board. This risk has also abated over time, yet remains present in some hazard reports. SR&QA personnel are conducting workshops with project safety engineers to educate them further on describing software risk. We have also provided a two-page "user guide" with examples of how safety engineers can specify software causes of hazards that has been well-received by SR&QA personnel. Furthermore, NASA technicians are considering changes to the hazard tracking system to enable safety engineers to mark software causes, controls and verifications as involving software.

3) ***Risk 3*** *– "Lumped" software causes and controls impede verification.*

Many hazard reports placed all software causes and most software controls under a single cause labeled "Software-based error." In many cases, this cause had a single control with multiple pages of software design and operational information. This large control then had a single verification. This single control, while highly detailed, presents risk in that software design and behaviors will not be individually verified.

As with the previous risk, CSERP and SR&QA personnel are working closely with project safety engineers to "modularize" the description of software causes controls instead of treating software as a single black-box entity. A constant challenge faced by CSERP, SR&QA and safety engineers is determining when differentiating complex hardware and software functionalities into multiple causes and controls is appropriate. Complex causes and controls introduce risk that some individual risks may not be well understood. However, creating controls also entails significant additional verification effort that may yield little return if the cause/control was largely covered elsewhere.

4) ***Risk 4*** *– Incorrect references to hazard reports, causes and controls impair traceability.*

A number of references to missing or incorrect hazard reports, causes or controls were observed. The most substantial risk is that a cause may not be adequately controlled when one or more of its controls are transferred to an incorrect or missing hazard report, cause, or control. NASA technicians are currently deploying improved functionality in the HTS to allow safety engineers to create

explicit references to other hazards, causes, controls and verifications in the hazard reports. This functionality will be backed by automated verification and bookkeeping.

5) ***Risk 5*** *– Sub-controls dissuade independent verification and add overhead.*

Many HRs have controls that contain enumerated "sub-controls." Greater confidence in the control may be gained by verifying the sub-controls independently. Furthermore, additional risk is introduced in that references to sub-controls may become lost or incorrect as these references must necessarily be manual instead of taking advantage of the technology available in the hazard tracking system. As in Risk 3, CSERP and SR&QA personnel are working closely with safety engineers to determine the best methods for separating out and managing the overhead associated with complex controls.

6) ***Risk 6*** *– Ubiquity of transferred causes and controls may mask software risk.*

Across the projects, 23-31% of causes and 11-22% of controls were transferred. While necessary and appropriate in documenting hazards, transferred causes and controls represent added risk. The applicability of transferred causes and the adequacy of transferred controls must be re-evaluated in their original context whenever any changes are made to the causes or controls. Furthermore, additional bookkeeping is necessary to ensure that the references to hazard reports, causes and controls are up to date (see Risk 4). Transferred causes and controls also make it difficult to understand the impact of software.

Stronger tool support (as described in Risk 4) enables better traceability and bookkeeping, but also enables analysis that can be used to quantify software risk. Coupled with marking causes and controls as software (as described in Risk 2), the HTS tool could then report comprehensively the number of software causes and controls by automatically resolving dependencies between hazards.

VI. Summary and Future Work

We analyzed 154 hazard reports from the preliminary design phases of three flight hardware projects in the NASA Constellation program. Our goals were to: 1) quantify the prevalence of software in hazards, causes and controls; and 3) to evaluate software causes of hazardous conditions according to their specificity. We found that 49-70% of hazardous conditions in the three systems could be caused by software or software was involved in the prevention of the hazardous condition. We also found that 12-17% of the 2013 hazard causes involved software, and that 23-29% of all causes had a software control. Furthermore, 10-12% of all controls were software-based.

By analyzing hazard reports, we gained insight into risk areas within the software safety analysis process by analyzing its process artifacts. We identified six risks in software safety analysis reporting. We are working with NASA SR&QA personnel in an ongoing effort to educate NASA safety engineers on describing software safety risk, to improve NASA process documents and training materials, and to provide tool support to the software safety process.

In the future, we are planning to compare the various systems in an attempt to build baselines for the various software measures. This will allow us to interpret the data more effectively. For example, if the three systems are similar, then we might expect software to play a similar role in the causes and controls. If not, how might we characterize the differences? Analysis of the data shows that the software related hazards, causes, and controls for project B are much lower than those for project A. Why might this be true? The two systems may be sufficiently different with respect to their use of software, or the incompleteness of the data and the numerous transfers may be masking their similarities.

A longer term goal is to evaluate multi-system, "integrated" hazards. It may be difficult to consistently measure the software scope in hazard reports among subsystems because of difference in reporting software causes and controls (software's role), i.e., a sufficient software risk assessment across the program is difficult, expensive, or maybe even impossible. Finally, we plan to continue our evaluation to hazard reports on other NASA systems and extend our evaluations to include controls and verifications.

Acknowledgment

This research was supported by NASA OSMA SARP grant NNX08AZ60G to the Fraunhofer CESE. We would like to acknowledge the help of Karen Fisher and Risha George at NASA Goddard Space Flight Center for providing us support and access to people and artifacts of the Constellation Program.

References

[1] ARIANE 5 Flight 501 Failure, Report by the Inquiry Board, Paris, July 19, 1996,. http://esamultimedia.esa.int/docs/esa-x-1819eng.pdf.

[2] V. Basili and D. Weiss, "A Methodology for Collecting Valid Software Engineering Data," *IEEE Transactions on Software Engineering*, vol.10(3), Nov. 1984, pp. 728-738.

[3] V. Basili, F. Marotta, K. Dangle, L. Esker, and I. Rus, "Measures and Risk Indicators for Early Insights Into Software Safety," *Crosstalk*, Oct. 2008, pp. 4-8.

[4] N. G. Leveson and C. S. Turner, "An investigation of the *Therac-25* accidents," *IEEE Computer*, 26(7), July 1993, pp.18-41, doi:10.1109/MC.1993.274940.

[5] B. Prasad, Concurrent Engineering Fundamentals – Integrated Product and Process Organization, Prentice Hall, Upper Saddle River NJ, 1996.