20 Aug 2007

Risk Assessment in Early Software Design Based on the Software Function-Failure Design Method

Jayson P. Vucovich

Robert B. Stone
Missouri University of Science and Technology

Xiaoqing Frank Liu
Missouri University of Science and Technology, fliu@mst.edu

Irem Y. Tumer

Recommended Citation

J. P. Vucovich et al., "Risk Assessment in Early Software Design Based on the Software Function-Failure Design Method," *Proceedings of the 31st Annual International Computer Software and Applications Conference, 2007*, Institute of Electrical and Electronics Engineers (IEEE), Aug 2007.
The definitive version is available at https://doi.org/10.1109/COMPSAC.2007.184

Risk Assessment in Early Software Design Based on the Software Function-Failure Design Method

Jayson P. Vucovich and
Robert B. Stone
Interdisciplinary Engineering
University of Missouri - Rolla
Rolla, Missouri 65409
{jayson, rstone}@umr.edu

Xiaoqing (Frank) Liu
Computer Science
University of Missouri - Rolla
Rolla, Missouri 65409
fliu@umr.edu

Irem Y. Tumer
Mechanical, Industrial, and Manufacturing Engineering
Oregon State University
Corvallis, OR 97331-6001
irem.tumer@oregonstate.edu

Abstract

Potential software failures present a sizable risk element in the design and development of many systems. In this paper, we augment the Software Function-Failure Design method, which is capable of predicting potential software failures in the very early stages of design, with the Risk in Early Design technique. This synergistic combination allows a risk assessment to be conducted at an early time in the software development process when traditional techniques are not applicable. The results are concise risk statements regarding the potential failure of functionalities with likelihood and consequence quantifications that can be used as part of a risk management program. The process is illustrated using a software failure database for the NASA Mars Exploratory Rover.

1. Introduction and motivation

The potential for software failures presents a sizable risk element in any software-intensive system. Where long-term planning, complex designs, autonomous control, and expensive hardware are tightly integrated—as in the NASA Mars missions—a single software failure can cause the loss of a $328 million project. In the case of manned missions, the loss can be incalculable.

Software failures present an increasing proportion of risk. The reliability of design and manufacture for mechanical and electromechanical components has increased to such a level that software now constitutes the plurality of failures in many hardware-software hybrid systems—33.9% in the case of one NASA Spacecraft [11].

For these reasons, it is critical to incorporate risk analysis into every aspect of the design, implementation, and execution of such missions. With the pervasiveness of software and software-hardware hybrid systems, risk analysis is beneficial and often necessary even for corporate endeavors and smaller scale projects. Not only must risk be ascertained, but steps must be taken to prevent or identify potential failures in software.

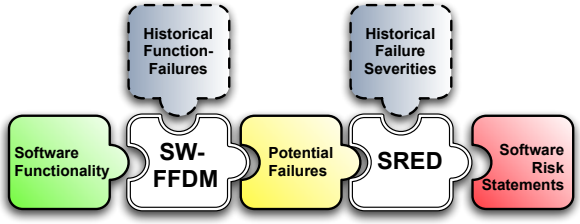

Figure 1. Schematic Overview of Using FFDM and RED to Assess Software Risk

This paper introduces SRED (Software Risk in Early Design), a method to be applied early in the software design process to identify and analyze the risk presented by potential software failures. SRED is built upon previous efforts to transform the Function-Failure Design Method [15, 16] from an electromechanical design domain to the software design domain. With the *Software Function-Failure Design Method* developed, this paper adapts and demonstrates the corresponding *Risk in Early Design* (RED) method [3, 4] to the software domain, to provide a software risk assessment technique based upon functionality—often the only information available in the very early design stages. Software functionality is used with historical function-failure data to generate potential failures via SWFFDM. These potential failures are combined with historical failure severity information to generate a software risk assessment via SRED, as illustrated in Figure 1.

SRED allows the early assessment of risk, which can

31st Annual International Computer Software and Applications Conference(COMPSAC 2007)
0-7695-2870-8/07 $25.00 © 2007 IEEE

guide future, more-detailed risk assessment techniques, provide a test-case development guide, and assist in making the decision on whether or not a software product has been tested enough to be safely released. By leveraging the advantages of the *Function-Failure Design Method* and *Risk in Early Design*, SRED eliminates many shortcomings of applying traditional engineering-based risk assessment techniques (*e.g.*, FMEA) to the early design stages of software development (Section 2.1). Furthermore, SRED is applicable in the early stages of design, allowing the application of continuous risk management at a time when traditional techniques cannot be successfully applied.

This paper proceeds by describing current software risk assessment techniques, the background and application of the Function-Failure Design Method (FFDM) to software, and an overview of the Risk in Early Design (RED) method for FFDM in the *Related work* section. Next, Section 3 illustrates the development of RED into SRED using the NASA Mars Exploratory Rover as an example of its application to the software development process. Section 4 demonstrates how the risk assessment provided by SRED can be used to control risk using the analysis as a decision-making tool when releasing a tested software product. Finally, concluding remarks and future work are discussed in Section 5.

2. Related work

2.1. Current software risk assessment techniques

Due to the complex and often critical nature of software in many systems—from nuclear reactors to consumer products—many techniques for risk assessment and analysis have been inspired by or adapted from the more developed engineering domain [10]. Chiefly among them is the Failure Modes, Effects, and Criticality Analysis (FMECA/FMEA), developed by the aerospace and automotive industries [10]. FMEA is a bottom-up approach to identify potential failures in a product, assess their hazard, and identify mitigation steps. From these assessments, a risk number is determined for each potential failure.

FMEA has been applied to software (leading to SFMEA) as early as 1983 [2]. SFMEA is frequently combined with Fault-Tree Analysis and has been performed successfully on a wide range of systems, including spacecraft flight software. SFMEA is the *de facto* method of risk assessment for software [1, 7, 8].

Despite its importance and wide use, SFMEA is subject to several shortcomings. To be meaningful, a SFMEA must be performed late in the design cycle when the software product is available to be examined in detail. This limits its usefulness in early design when a risk assessment can have the most impact with the least penalty for change [2, 8]. Because the assessment is manual and dependent on soliciting assessments from experts, a team of engineers intimately familiar with the design and implementation of the software must gather to perform the tedious task of producing the subjective risk assessment [8]. Pentti, in conducting an extensive literature survey on the suitability of SFMEA, concludes that, "A common opinion is that FMEA is applicable to software-based systems only to a limited extent" [10].

Many other methods exist for assessing risk and discovering latent software failures in the early and late stages of design as well as during development. Sophisticated techniques assess requirements completeness, perform static code analysis, aggregate software metrics, or even formally prove correctness [6]. Each of these methods provides a detailed analysis of risk or correctness using a level of design detail not always available in the earliest stages. Some, such as SFMEA can be stimulated with the results of SWFFDM.

2.2. The Software Function-Failure Design Method

A clear need exists to incorporate risk assessment as early in the development process as possible. The early availability of risk information allows improvements to be made to the design at a stage when the penalty for change is minimal. This is true both for mechanical and software based systems and was a driving realization for the development of the Function-Failure Design Method for electromechanical systems [15, 16]. Prior work by the authors has adapted the FFDM to software.

In the Software Function-Failure Design Method (SWFFDM), historically observed failures of software are classified by a failure mode taxonomy and catalogued by the functionality provided by the failed software module. This catalog links failure to functionality through components. Components may be software modules, libraries, or other separately identifiable implementations of software. When a new software product is designed, its intended functionality can be used to generate a set of potential failures. This process is automatic and can be carried out by a novice software developer with only top-level knowledge of the functionality of the product under analysis. Because SWFFDM is based on historical data, it is not dependent upon the developer to think of all the possible failure modes in order to produce results.

The key components of the FFDM are the functionality taxonomy, the failure-mode taxonomy, and the failure knowledge-base, which takes the form of a matrix. In order to populate this matrix, the functionality taxonomy is used to capture the functionality provided by components in a consistent manner. This is captured in a *Function-Component* (**EC**) matrix that correlates functionality to

specific implementations. Next, the failure-mode taxonomy is used to consistently capture observed failures and correlate them to the specific, failed implementations. This is termed a *Component-Failure* (**CF**) matrix. These two matrices are multiplied together to yield a *Function-Failure* (**EF**) matrix, which correlates failure modes with functionality and comprises the failure knowledge-base. This is expressed in Equation (1).

$$\begin{aligned} \mathbf{EF} &= \mathbf{EC} \times \mathbf{CF} \\ \mathbf{EC}_{k,i} &= \begin{cases} 1 & \text{Where component } i \text{ has been used} \\ & \text{to provide functionality } k. \\ 0 & \text{Otherwise} \end{cases} \\ \mathbf{CF}_{i,j} &= n, \text{ where component } i \text{ has experienced} \\ & \quad n \text{ failures of type } j. \\ \mathbf{EF}_{k,j} &= p, \text{ where functionality } k \text{ is associated} \\ & \quad p \text{ times with failure mode } j. \end{aligned} \tag{1}$$

Potential failure modes can be identified for a new project by consulting the failure knowledge-base represented in the **EF** matrix. Functionality is represented in rows, while failure modes are represented in columns. Since failure is related to functionality through components by matrix multiplication (Equation (1)), the failure knowledge-base contains predictions of failure for functionality based upon historically observed failures of components with similar functionality.

2.3. The Risk in Early Design (RED) method

While the SWFFDM provides a list of potential failures of functionality present in a design, it does not provide any information about risk. The only way to differentiate among generated failure modes is by relative frequency of observation. To be truly useful, risk information must be associated with each potential failure to determine the relative amount of effort to be expended on discovering, preventing, or handling the given potential failures. For this reason, the Risk in Early Design (RED) method was developed [3, 4].

RED seeks to effectively communicate risk elements during the early stages of design—where the greatest opportunity for risk mitigation occurs. Furthermore, RED seeks to accomplish this without the need for a team of experts, the tedium, nor the subjectivity of language, which are present in many risk assessment techniques. In order to accomplish this, RED leverages FFDM to generate potential failures but augments the results with likelihood and consequence data. These two additional pieces of information, combined with their associated failure of functionality, constitute a risk element which conveys what Rosenberg, *et al*. term a *risk statement*, concisely communicating what can fail, how severe the failure would be, and a relative strength of potential for observing the failure [4, 13].

2.3.1 Likelihood and consequence mappings

Likelihood data is determined in relation to other potential failures from the Function-Failure matrix. Consequence data is determined from the addition of a $\mathbf{CF}'$ matrix. This matrix contains failure severity entries from a severity rating on a scale of increasing severity from 1–5. This scale is divided such that a severity entry of 1 implies the failure is relatively unnoticed by the casual observer and a severity entry of 5 implies a safety risk. For entries in $\mathbf{CF}'$ where no failure is recorded, the value 0 is used [4].

RED provides two methods for calculating likelihood (*L1-prod* and *L2*) and two methods for calculating severity (*C1* and *C2*) [4]. *L1-prod* provides a likelihood mapping which normalizes the likelihood of a failure to a scale of increasing likelihood of 0–5 based on the maximum value of entries in a subset of the **EF** matrix containing only functionality found in the product under analysis:

$$\mathbf{EF}_{relevant} = \mathbf{EF}_{k,j} \forall\, k \in \{\text{product functionality}\}. \tag{2}$$

The *L2* mapping provides the same normalization scale but normalizes against the maximum value of all entries in the **EF** matrix. Equation (3) provides the formulation, which depends on a specialized rounding function, int, that rounds *half-up* normally but rounds non-zero numbers less than 1 up to 1 [4].

$$\begin{aligned} \mathbf{L}_{k,j} &= int\left(5 \times \frac{\mathbf{EF}_{k,j}}{\max(\mathbf{EF})}\right) \\ \max(\mathbf{EF}) &= \begin{cases} \max(\mathbf{EF}_{relevant}) & \text{Mapping L1} \\ \max\limits_{k,j}(\mathbf{EF}) & \text{Mapping L2} \end{cases} \end{aligned} \tag{3}$$

The two consequence mappings similarly produce consequence values for each function-failure entry. The *C1* mapping provides a conservative estimate of consequence by returning the maximum severity of a failure for a given functionality. Alternatively, the *C2* mapping provides an augmented average severity among observed failures for a given functionality. The mappings are given in Equation (4) for *C1* and Equation (5) for *C2*. The formulation for *C2* depends on a value h, which represents the relevant number of function-failure combinations for that function (*i.e.,* the number of products in $\mathbf{EC}_{k,r} \times \mathbf{CF}'_{r,j}$ that are non-zero) [4].

$$\mathbf{C}_{k,j} = \max_{r}\left(\mathbf{EC}_{k,r} \times \mathbf{CF}'_{r,j}\right) \tag{4}$$

$$\mathbf{C}_{k,j} = int\left(\frac{1}{h}\sum_{r}\mathbf{EC}_{k,r} \times \mathbf{CF'}_{r,j}\right) \quad (5)$$

2.3.2 Choosing the appropriate mapping combination

Each combination of likelihood and consequence mappings serves a specific purpose. The *L1-prod* likelihood mapping provides a conservative estimate of risk but it can provide overly-high likelihood projections that may not be fully supported by the data. The *L2* mapping avoids this but can downplay underrepresented failures. The *C1* consequence mapping provides a conservative severity estimate but can be dominated by a single severe occurrence. The *C2* consequence mapping reduces the domination of a single high-severity occurrence without diluting the average, but it is more sensitive to the specific combinations of failures and severities recorded in the database [3].

These combinations give rise to a heuristic for selecting a consequence and likelihood mapping: Because of the conservative estimate of *C1*, it is better suited when the product involves human safety. If human safety is not a factor, *C2* often provides a more realistic assessment of risk. For system-level design situations, the *L2* likelihood mapping is most applicable since it is normalized against a wider range of failures. *L1* is more suited for subsystem design situations due to the more limited scope of normalization [3].

2.3.3 Using likelihood and consequence mappings

After selecting the appropriate combination of mappings, the data is summarized through the use of a fever chart. A fever chart is a 5-by-5 matrix that shows consequence on the horizontal axis and likelihood on the vertical axis. Each cell of the matrix displays the number of elements falling into that consequence-likelihood combination. The chart is color-coded such that the lower-left area is green to indicate low risk, the middle area is yellow to indicate moderate risk, and the upper-right area is red to indicate high risk [9]. Example fever charts can be seen in Section 3.3.

The fever chart is populated with the failure data produced by the FFDM procedure, plotted according to the associated values for the consequence and likelihood—taken from their appropriate mapping matrices. This combination of information provides an effective means of communicating identifiable risk elements. A risk statement is composed of the design parameter that fails (the functionality), the manner in which it may fail (the failure mode), the likelihood of failure, and the consequence. Using this information, potential failures can easily be divided among low, moderate, and high risks without introducing undue subjectivity.

2.4. Continuous risk management

Rosenberg, *et al*. present a model of continuous risk management in use at NASA's Software Assurance Technology Center (SATC). This model provides a circular, six-step continuous model for managing risk, beginning with *identification*. In this first step uncertainties are transformed into describable, distinct risks. In the second step, *analyze*, risk information is converted into decision-making information. Next, the *planning* step provides an opportunity to set actions in response to risk decisions. Step four provides for *tracking*, in step five tracking information is acted upon to *control* risk, while the final step is to *communicate and document* risk and mitigation for future access and awareness [13].

We will continue by stepping through the SRED process on an example project. At each step, we will discuss how SRED can be used as part of a continuous risk management model to provide early *identification* of risk elements. This early identification will then be used to *analyze* the risk information such that *planning* can be carried out in an informed state. We will also see how SRED contributes to risk *tracking*, *control*, and *communication*.

3. Developing SRED: Using the RED method with software

A model of the functionality for the software under assessment is used to identify potential failures via SWFFDM. Each potential failure is then fed into SRED to compute consequence and likelihood values. Thus, quantifiable, distinct risk statements are generated, which can be used for further analysis, as discussed in later sections.

The example project used to illustrate the adaptation of RED to software and for the use of SRED is the NASA Mars Exploratory Rover (MER). This mission had many software and software-hardware hybrid components. The relevant modules we will be focusing on are the star scanner orientation system, the robotic instrument arm for the Mars rovers, and the cruise stage craft itself. The star scanner data is actually comprised of several implementations of the star scanner module over many missions. The robotic arm (Instrument Deployment Device) holds several scientific instruments with specific software to drive their operation. The cruise stage craft (referred to as the *vehicle* in this paper) contains flight software originally developed for the Mars PathFinder mission. The software was reused in the Deep Space One mission and ultimately modified for use with MER.

The actual software was unavailable for examination; however a set of previously collected Problem and Failure Reports (PFRs) spanning several projects over several years

were available. The Problem and Failure Reporting system is maintained by NASA's Jet Propulsion Laboratory and contains actual failure data for unmanned spacecraft. Over 300 missions are represented, though only a small fraction of the reports were available to the authors [12]. From this limited set, software failures were extracted and identified as being part of a star scanner, the MER robotic instrument arm, or the MER vehicle. The data set is limited and incomplete, though it is enough to serve as an illustration of the SRED development and application process.

These software-related PFRs were used to develop a high-level software functionality taxonomy for embedded software. This was combined with a software failure taxonomy developed by Li, Smidts, *et al*. [6, 14] that was found to be applicable as mentioned in Section 2.2.

3.1. Constructing a function-failure knowledge-base

The key to the success of traditional FFDM, RED, and SRED alike is a quality failure knowledge-base as explained in Section 2. The quality of the predictions made are directly linked to the quality of captured historical failure data. The initial requirement of SRED is for a failure knowledge-base to be available.

Due to the incompleteness and availability of data, the PFRs for the star scanner and robotic arm system were used to develop the knowledge base, while the PFRs for the MER vehicle were set aside and will be used later as the example system under development to be analyzed using SRED. This allows the analysis provided by SRED to be compared with actual, discovered errors and for the illustration of its application to various stages of development. The PFRs are distributed among the three systems: Robotic Arm (4), Star Scanner (24), and MER Vehicle (11).

Each software PFR for the robotic arm and star scanner was manually examined to extract the failure and classify it in the failure mode taxonomy using the available description of what the error was and how it was ameliorated. Similarly, the functionality was determined using the same descriptions, classified, and combined to build up a failure knowledge base by filling in $\mathbf{EC}$ and $\mathbf{CF}$ matrices to arrive at an $\mathbf{EF}$ matrix as described in Equation (1). Furthermore, each PFR also records a failure severity as determined by the discovering engineer. This severity is used to develop a $\mathbf{CF}'$ matrix as described in Section 2.3.1.

The development of these failure and severity knowledge-bases is a necessary precondition for performing the risk analysis. Though it is the prerequisite step in the process, it is actually part of the last step of *communication and documentation* in the risk management model presented by Rosenberg, *et al*. As failures are encountered, part of the *tracking* step is to incorporate them into a repository that can be used to manage risk.

3.2. Performing an early risk evaluation: Identifying risk

The $\mathbf{EC}$ and $\mathbf{CF}$ matrices, their product (the $\mathbf{EF}$ matrix), and the $\mathbf{CF}'$ matrix, created using PFR data from the robotic arm and star scanner, are shown in Figures 2–5. These represent our repository of failure information based upon historic observations (or expert opinion in the case of severity) and will be used to analyze the MER vehicle design.

Supposing the MER vehicle software is in the early stages of development, only the desired functionality will be known. For SRED this functionality is expressed in the software functionality taxonomy but can be extracted from existing requirements documents. Due to the comparatively homogenous and abstract nature of available data, the desired functionality for the MER vehicle software is identical to the set of functionalities for which failures have been captured and corresponds to the rows of the $\mathbf{EF}$ matrix (Figure 4).

For our test example of the MER vehicle, we now have a set of potential failures based on the intended functionality as represented in Figure 4. Reading the first line of the matrix indicates that *Analyze Linear Data* can possibly fail by *incorrect attribute realization*, *de-synchronization*, *environment induced failure*, *etc*. Each of these failure modes has a specific meaning detailed in other work [6, 14]. Reading down the rows gives similar failures for other functionalities. The number in the corresponding cells indicates the relative support of the historical data for the potential of the given functionality failing by the given mode.

At this point, we have identified only what can fail and how through SWFFDM; we have not identified any risk. The next step in the SRED process is to apply RED where SWFFDM left off. Using the formulations provided in Equations (3) through (5), we can attach likelihood and consequence data to our potential failures and arrive at identifiable risks.

The likelihood mapping for the MER vehicle is given in Figure 6 according to Equation (3). Due to the limited amount of data, *L1* and *L2* are identical: the maximum entry in the complete $\mathbf{EF}$ matrix is the same as the maximum entry in the filtered matrix. Ordinarily, an *L1* mapping would be most appropriate as the MER vehicle is a subsystem of a larger system.

Similarly, a consequence mapping is created according to Equation (4) or (5). Because the MER vehicle is unmanned, the less-conservative *C2* mapping (Figure 7) is more appropriate. With these three elements ($\mathbf{EF}$, $\mathbf{L1}$, $\mathbf{C2}$) combined, risk items have been identified and quantified. A risk statement can be constructed by matching the data from

SS & RA Function-Component Matrix (Bin)	Attitude Control	Attitude Control Interface	Microimager Camera Interface	Operating System	Panoramic Camera Interface	Radar Altimeter	Star Scanner Analyzer	Star Scanner System	X-Ray Spectrometer System
Analyze Linear Data	1	0	0	0	0	1	1	0	1
Analyze Scenes	0	0	0	0	0	0	1	0	0
Approximate Solutions	1	0	0	0	0	1	1	0	1
Control Analog Device	0	1	0	0	0	1	1	0	1
Control Digital Device	1	0	1	1	1	0	0	1	0
Generate Graphics	0	0	1	0	1	0	0	0	0
Manage Hardware	0	0	1	1	1	0	0	0	0
Manage Processes	0	0	0	1	0	0	0	1	0
Manage Reliability	1	1	0	1	0	0	0	1	0
Perform Arithmetic Calculations	1	1	0	1	0	1	1	1	1
Read or Write Data	1	1	1	1	1	1	1	1	1

Figure 2. Star Scanner and Robotic Arm Function-Component ($\mathbf{EC}$) Matrix

SS & RA Component Failure	Attitude Control	Attitude Control Interface	Microimager Camera Interface	Operating System	Panoramic Camera Interface	Radar Altimeter	Star Scanner Analyzer	Star Scanner System	X-Ray Spectrometer System
Attitude Control	0	0	0	0	0	0	0	0	0
Attitude Control Interface	0	0	0	0	0	0	0	0	0
Microimager Camera Interface	1	0	0	0	0	0	0	0	0
Operating System	1	2	1	0	1	0	2	1	0
Panoramic Camera Interface	0	1	0	0	0	0	1	1	0
Radar Altimeter	0	0	0	0	1	0	0	0	0
Star Scanner Analyzer	1	0	0	0	0	0	2	0	0
Star Scanner System	0	1	1	1	0	1	0	0	1
X-Ray Spectrometer System	0	1	0	0	0	0	0	0	0

SS & RA Component Failure	Deadlock	Environ.-Induced Failure	Impl. Unspecified Function	Incorr. Attrib realization	Incorr. Funct Realization	Lockout	De-Synchronization	Function:Omission	Load:Overload	Range:Out of Range	Rate:Too Fast	Rate:Too Slow	Value:Incorrect Value
Attitude Control	0	0	0	0	0	0	0	0	0	0	0	0	1
Attitude Control Interface	0	0	0	0	0	0	0	0	0	0	1	0	0
Microimager Camera Interface	1	0	0	0	0	0	0	0	0	0	0	0	0
Operating System	1	2	1	0	1	0	2	1	0	1	0	0	0
Panoramic Camera Interface	0	1	0	0	0	0	1	1	0	0	0	0	0
Radar Altimeter	0	0	0	0	1	0	0	0	0	0	0	0	0
Star Scanner Analyzer	1	0	0	0	0	0	2	0	0	0	0	1	1
Star Scanner System	0	1	1	1	0	1	0	0	1	0	0	0	1
X-Ray Spectrometer System	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 3. Star Scanner and Robotic Arm Component-Failure ($\mathbf{CF}$) Matrix

SS & RA Function Failure	Deadlock	Environ.-Induced Failure	Impl. Unspecified Function	Incorr. Attrib realization	Incorr. Funct Realization	Lockout	De-Synchronization	Function:Omission	Load:Overload	Range:Out of Range	Rate:Too Fast	Rate:Too Slow	Value:Incorrect Value
Analyze Linear Data	1	1	0	0	1	0	2	0	0	0	0	1	2
Analyze Scenes	1	0	0	0	0	0	2	0	0	0	0	1	1
Approximate Solutions	1	1	0	0	1	0	2	0	0	0	0	1	2
Control Analog Device	1	1	0	0	1	0	2	0	0	0	1	1	1
Control Digital Device	2	4	2	1	1	1	3	2	1	1	0	0	2
Generate Graphics	1	1	0	0	0	0	1	1	0	0	0	0	0
Manage Hardware	2	3	1	0	1	0	3	2	0	1	0	0	0
Manage Processes	1	3	2	1	1	1	2	1	1	1	0	0	1
Manage Reliability	1	3	2	1	1	1	2	1	1	1	1	0	2
Perform Arithmetic Calculations	2	4	2	1	2	1	4	1	1	1	1	1	3
Read or Write Data	3	5	2	1	2	1	5	2	1	1	1	1	3

Figure 4. Star Scanner and Robotic Arm Function-Failure ($\mathbf{EF}$) Matrix

each entry and is more readily communicated if listed out.

3.3. Analyzing risk

With risk statements created for the MER vehicle, we move on to the task of analyzing the risk data to convert it into a decision-making guide. At this stage, 95 individual risk statements are available, ranging from the practically inconsequential to the severe. Some items must be given priority consideration over others.

To accomplish this task, we employ the concept of a fever chart as mentioned in Section 2.3.3. A fever chart allows the 95 risk statements to be condensed into low, medium, and high risk segments. This is done by simply plotting each risk statement according to the $(consequence, likelihood)$ coordinates given by the $\mathbf{C}$ and $\mathbf{L}$ matrices. This has been done to produce the fever charts shown in Figure 8 for the *C2* mapping. For illustrative purposes, the more conservative *C1* fever chart is also shown in Figure 9.

The fever chart in Figure 8 shows that a majority of the potential failures fall into the low-risk zone, a few are in the moderate-risk zone, and none are in the high-risk zone. Visually, the chart conveys that while a few moderate-risk failures are predicted, the overall risk is low. This chart effectively communicates that *Control Digital Device* failing due to *Overload* $(1, 1)$ is not as risky as the potential for *Control Digital Device* failing due to *De-Synchronization* $(3, 4)$.

RED provides another means of conveying overall risk. The center of risk ($\overline{r}$) can be computed as the weighted average of the consequence and likelihood. This is analogous to calculating the center of mass for a fever chart [4].

3.4. Applying the risk analysis: Planning

Because the MER vehicle shows a very low center of risk ($\overline{r}(C2L1) = (1.56, 1.60)$), it may be appropriate to accept many of the risks as presented while making some plans to mitigate the moderate risks. Because of the quality of this illustrative failure knowledge-base, low risk potential failures should not be discounted: while there is little evidence to support counting them as significant risks, there is also little evidence to support discounting their risk. Ideally, the failure knowledge-base would draw on a wider range and depth of historical failures.

One clear mitigation practice that can be guided by the SRED results is simply to create test cases that are sure to discover failures in the indicated functionalities. For example, the moderate risk presented by the functionality *Control Digital Device* failing due to *De-Synchronization* $(3, 4)$ could be mitigated by ensuring that test cases for MER vehicle modules that participate in controlling a digital device

SS & RA CF-Prime	Incorr. Attrib realization	De-Synchronization	Deadlock	Lockout	Environ.-Induced Failure	Impl. Unspecified Function	Incorr. Funct Realization	Function:Omission	Load:Overload	Range:Out of Range	Rate:Too Fast	Rate:Too Slow	Value:Incorrect Value
Attitude Control	0	0	0	0	0	0	0	0	0	0	0	0	1
Attitude Control Interface	0	0	0	0	0	0	0	0	0	0	1	0	0
Microimager Camera Interface	3	0	0	0	0	0	0	0	0	0	0	0	0
Operating System	1	3	2	0	1	0	2	1	0	1	0	0	0
Panoramic Camera Interface	0	1	0	0	0	0	1	1	0	0	0	0	0
Radar Altimeter	0	0	0	0	2	0	0	0	0	0	0	0	0
Star Scanner Analyzer	2	0	0	0	0	0	2	0	0	0	0	2	1
Star Scanner System	0	2	3	1	0	1	0	0	1	0	0	0	2
X-Ray Spectrometer System	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 5. Star Scanner and Robotic Arm Component-Failure Severity (CF') Matrix

Vehicle Likelihood - L1/L2	Incorr. Attrib realization	De-Synchronization	Deadlock	Lockout	Environ.-Induced Failure	Impl. Unspecified Function	Incorr. Funct Realization	Function:Omission	Load:Overload	Range:Out of Range	Rate:Too Fast	Rate:Too Slow	Value:Incorrect Value
Analyze Linear Data	1	1	0	0	1	0	2	0	0	0	0	1	2
Analyze Scenes	1	0	0	0	0	0	2	0	0	0	0	1	1
Approximate Solutions	1	1	0	0	1	0	2	0	0	0	0	1	2
Control Analog Device	1	1	0	0	1	0	2	0	0	0	1	1	1
Control Digital Device	2	4	2	1	1	1	3	2	1	1	0	0	2
Generate Graphics	1	1	0	0	0	0	1	1	0	0	0	0	0
Manage Hardware	2	3	1	0	1	0	3	2	0	1	0	0	0
Manage Processes	1	3	2	1	1	1	2	1	1	1	0	0	1
Manage Reliability	1	3	2	1	1	1	2	1	1	1	1	0	2
Perform Arithmetic Calculations	2	4	2	1	2	1	4	1	1	1	1	1	3
Read or Write Data	3	5	2	1	2	1	5	2	1	1	1	1	3

Figure 6. MER Vehicle Likelihood ($L1/L2$) Mapping

Vehicle Consequence - C2	Incorr. Attrib realization	De-Synchronization	Deadlock	Lockout	Environ.-Induced Failure	Impl. Unspecified Function	Incorr. Funct Realization	Function:Omission	Load:Overload	Range:Out of Range	Rate:Too Fast	Rate:Too Slow	Value:Incorrect Value
Analyze Linear Data	2	1	0	0	2	0	2	0	0	0	0	2	1
Analyze Scenes	2	0	0	0	0	0	2	0	0	0	0	2	1
Approximate Solutions	2	1	0	0	2	0	2	0	0	0	0	2	1
Control Analog Device	2	1	0	0	2	0	2	0	0	0	1	2	1
Control Digital Device	2	2	3	1	1	1	2	1	1	1	0	0	2
Generate Graphics	3	1	0	0	0	0	1	1	0	0	0	0	0
Manage Hardware	2	2	2	0	1	0	2	1	0	1	0	0	0
Manage Processes	1	3	3	1	1	1	2	1	1	1	0	0	2
Manage Reliability	1	3	3	1	1	1	2	1	1	1	1	0	2
Perform Arithmetic Calculations	2	2	3	1	2	1	2	1	1	1	1	2	1
Read or Write Data	2	2	3	1	2	1	2	1	1	1	1	2	1

Figure 7. MER Vehicle Consequence ($C2$) Mapping

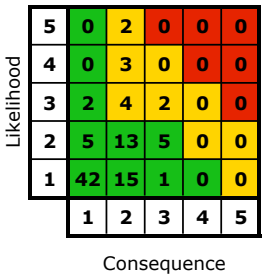

Figure 8. Fever Chart for the $C2L1$ Mapping

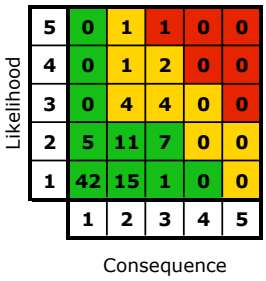

Figure 9. Fever Chart for the $C1L1$ Mapping

include specific tests for their synchronization with other processes or threads. This also improves the traceability of the project if test cases are tied to requirements through the risk analysis.

Depending both on the criticality of the software project and the center of risk presented by the SRED analysis, it may be beneficial to plan to perform a more detailed Probabilistic Risk Assessment (PRA) such as SFMEA. Generally performing a PRA requires generating a set of initiating events to transform into risk profiles. Clearly SRED can be of assistance by providing a set of potential risks that can be added to or subtracted from by a team of experts [3, 4]. With an established failure knowledge-base and functional description of the software project, the SRED analysis could be generated by a novice in a matter of minutes. Not only does this reduce tedium and speed such detailed PRA processes along, it also allows the experts to focus on the details of the analysis and the generation of mitigation recommendations.

4. Using RED to make the decision to release: Tracking and controlling risk

Moving along in our illustration of how SRED can be applied to the continuous risk management process, we arrive at the late stages of software development when testing is being conducted. After some amount of testing, either due to schedule constraints or other forces, the decision must be made of whether or not the software is ready to be released. This decision ought to be made with as much information on risk as possible.

Recalling the discussion of PFRs from Section 3.1 there were eleven software failures actually discovered during rigorous testing of the MER vehicle. If each of these failure discoveries were tracked in conjunction with the SRED analysis, we could continuously update the risk assessment by removing them from the list of potential failures once they are fixed. This would provide a dynamic

assessment that could be used to control exposure to risk.

A common technique used to assess risk at the end of development is error trending, which plots the cumulative number of errors discovered versus the cumulative number of days of testing. From this plot, various trending models can be fit to the data to predict the total number of errors present in the software. The designers can then determine how many errors remain to be discovered as well as how long their discovery may take. This information can be used to determine if the amount of risk associated with releasing the software with a few undiscovered errors is acceptable [5]. SRED can be used to augment this technique by providing more information about the risk associated with the remaining software errors.

5. Concluding Remarks

Widely used risk assessment and mitigation techniques such as static code analysis, software metric aggregation, and SFMEA provide an important, detailed assessment of risk in software projects when projects have progressed to the implementation stages and enough data is available to perform these analyses. To be truly effective, however, methods must be applied which incorporate awareness of risk in all stages of development. The early analysis and design stages present the unique opportunity to affect risk when changes can be made with the least impact to budget and schedule.

To fulfill the need for an easily applied risk assessment technique that can be used in the early stages of design—when detailed analysis cannot be carried out—the Software Function-Failure Design Method was augmented through the application of the Risk in Early Design technique. This paper has shown how RED was applied to the software domain to create SRED. NASA's Mars Exploratory Rover served as an example to illustrate how SRED can be incorporated as part of a continuous risk management program.

Acknowledgments

This work is supported by the NASA Ames Research Center under grant NCC 2-1380. Any opinions or findings of this work are the responsibility of the authors and do not necessarily reflect the views of the sponsors or collaborators.

References

[1] J. Bowles and C. Wan. Software failure modes and effects analysis for a small embedded control system. In *Proceedings of the Annual Reliability and Maintainability Symposium*, number 0-7803-6615-8/01. IEEE, 2001.

[2] P. Goddard. Software FMEA techniques. In *Proceedings of the Annual Reliability and Maintainability Symposium*, number 0-7803-5848-1/00, pages 118–123. IEEE, 2000.

[3] K. Grantham Lough, R. Stone, and I. Tumer. Prescribing and implementing the risk in early design (RED) method. In *Proceedings of DETC'06*, number DETC2006-99374, Philadelphia, PA, September 2006.

[4] K. Grantham Lough, R. Stone, and I. Tumer. The risk in early design (RED) method: Likelihood and consequence formulations. In *Proceedings of DETC'06*, number DETC2006-99375, Philadelphia, PA, September 2006.

[5] L. Hyatt and L. Rosenberg. Software metrics programs for risk assessment. In *Proceedings of the 47th International Astronautical Congress and Exhibition, 29th Safety and Rescue Symposium*, Beijing, China, October 1996.

[6] B. Li, M. Li, S. Ghose, and C. Smidts. Integrating software into PRA: A software related failure mode taxonomy. In *ISREE '03: Proceedings of the 14th International Symposium on Software Reliability Engineering*, pages 457–467, Washington, DC, USA, November 2003. IEEE Computer Society.

[7] R. R. Lutz and H.-Y. Shaw. Applying adaptive safety analysis techniques. In *10th International Symposium on Software Reliability Engineering*, number ISSRE.1999.809309, page 42. IEEE, 1999.

[8] R. R. Lutz and R. M. Woodhouse. Experience report: Contributions of SFMEA to requirements analysis. In *Proceedings of the ICRE*, number 0-8186-7252-8/96, pages 44–51. IEEE, 1996.

[9] L. Meshkat, S. Cornford, and T. Moran. Risk based decision tool for space exploration missions. In *Proceedings of the AIAA Space Conference*, number AIAA 2003-6377, Long Beach, CA, September 2003.

[10] H. Pentti and H. Atte. Failure mode and effects analysis of software-based automation systems. Technical Report STUK-YTO-TR 190, STUK, P.O.Box 14, FIN-00881 Helsinki, FINLAND, 2002.

[11] J. Quinn. Flight P/FRs and the design review process. Reliability Interim Significant Report RISR 16302-1, JPL D-11381, NASA Jet Propulsion Laboratory, 1994.

[12] R. Roberts, R. Stone, I. Tumer, and A. Brown. A function-based exploration of JPL's problem/failure reporting database. In *Proceedings of the 2003 ASME International Mechanical Engineering Congress and Expo*, number IMECE2003-42769, Washington, DC, USA, November 2003.

[13] L. Rosenberg, T. Hammer, and A. Gallo. Continuous risk management at NASA. In *Proceedings of Quality Week Conference*, San Francisco, CA, May 1999.

[14] C. Smidts, R. W. Stoddard, and M. Stutzke. Software reliability models: An approach to early reliability prediction. In *ISSRE '96: Proceedings of the 7th International Symposium on Software Reliability Engineering*, page 132, Washington, DC, USA, 1996. IEEE Computer Society.

[15] R. Stone, I. Tumer, and M. Stock. Linking product functionality to historic failures to improve failure analysis in design. *Research in Engineering Design*, 2005.

[16] R. Stone, I. Tumer, and M. Van Wie. The function-failure design method. *ASME Journal of Mechanical Design*, 2003.