

How FMEA Improves Hardware and Software Safety & Design Reuse

Nematollah Bidokhti
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134 USA
nbidokht@cisco.com

1.0 Introduction

Today's market needs and applications demand products with robust hardware and more importantly reliable and deterministic software.

Reliable software is defined as a piece of code, program or application that every time it is executed, it produces the same output and operates consistently. Deterministic software is defined as a software program that once it fails, it causes the same impact and is detected consistently.

Once the software is determined to be reusable for the desired application, it is important to perform the appropriate analysis to identify all possible failure modes associated with the design with respect to the new environment and its association with the components of the current architecture. One technique proven useful is the Failure Modes and Effects Analysis (FMEA).

In this paper we will discuss the following:

- Hardware and software relationships
- Design reuse challenges
- Use of Failure Modes and Effects Analysis (FMEA) as a remedy
- FMEA requirements and process
- When and who should perform FMEA
- What software component should go through FMEA

2.0 Hardware and Software Relationships

There can be situation where software faults are manifested as if they were hardware failures and cause the loss of output. Therefore, no longer can we decouple hardware from software in our approach to reuse of system architecture design and development. As the new technologies are developed, hardware and software are getting closer and software assumes a much more important role in the system operation.

The relationship between HW and SW FMEA is evident where both HW and SW FMEAs can be performed at functional, interface and detailed level.

HW and SW FMEA Correlation		
Types	Hardware	Software
Functional	Concept/Preliminary design phase	During top level software design
Interface	Physical interfaces between major system elements	Interfaces between dissimilar software and hardware elements
Detailed	Part level	Code Level

Therefore, hardware failure impacts software operation and vice versa. It must be noted that hardware failures are more understood than software and less dependent to other factors. There are many categories of software failures such as:

- Requirements
- Interface
- Fault Tolerance
- Resource Usage
- Data

These categories are further described in the section called "Software Failure Modes".

3.0 Design Reuse Challenges

For the past decade the market has pressured product manufacturers to reduce their product development life cycle significantly in order to decrease the time to market, releasing products with significantly new features.

This time to market requirement has raised the need for design reuse at a higher level. Therefore, hardware and software designers usually borrow or build from designs from past products.

In theory this is the right thing to do. It improves time to market and decreases product development cost. But, in fact if a low or marginally designed hardware or software is selected to be used on the new product, we carry the same issues to the new product and increase the probability of failures in addition to newly created issues.

No designer would like to use a design that is less than solid. But the time pressure to deliver the product within the allocated time forces them to select portions of

previous design without performing any risk management activity or researching the history such as test results or field/operation issues .

The risk increases or decreases based on the company’s development standard and designers’ skill level. In other words, if there is a clear development process where designers are required to perform detail design reviews, participate in peer code reviews and perform modeling such as Unified Modeling Language (UML), there is a higher probability that many design issues and corner cases have been identified. We will refer to this subject later in this paper.

Following are some of the challenges of design reuse:

Lack of documentation: There are many situations where the software designer is focused on coding, testing and delivering the software where documentation is overlooked. This issue creates a possible misunderstanding of the original intent of the software, and it increases the integration time.

Original designer no longer available: It is normal to see designers move to new project or leave the company for better opportunities. This will not be as significant if there is detailed code documentation. Lack of adequate documentation along with loss of the original developer will make the design reuse more challenging.

Change in requirements: The original code was developed to meet a set of requirements. Once any part of the requirements is modified, this will impact the code to some level. The extent depends on the complexity of the requirements and implementation method. FMEA could be used to highlight some of these issues.

Reliability goals: Generally set at the beginning of the project with respect to the architecture at hand. This could impact the code implementation methodology.

Performance objectives: Often there are clear performance objectives from Customers, documented in product requirements document. Design reuse could impact this metric based on the number of calls to different software entities and the order in which they are executed.

4.0 What is FMEA?

FMEA is a risk management activity that addresses product safety and reliability where its purpose is to identify and document all possible failure modes in hardware and or software. FMEA should not just be performed on the mission and safety critical system; as it

is a good risk analysis process that can be applied on a various types of systems such as Telecommunication systems. The purpose of the FMEA is not to measure the reliability of the product. Rather, it is a methodology that helps build a reliable design and fault management into the product.

How does FMEA impact design reuse? The main benefit is the ability to identify various failure modes and their impact to the system and provide a mechanism to quantify the risk of the selected software. Once this information is prepared and stored in the FMEA database, usage of this software by any designer at any point in time can be traced and understood to what are the risks that the user of the software needs to consider in their own application of the software.

FMEA is a powerful design risk analysis method which could be done at different levels of hardware such as ASIC, Board, System and Network level (as applicable) where each level provides a specific benefits. But the FMEA process level remains the same for the hardware and software. As it was mentioned earlier they are functional, interface and detailed.

Software functional FMEA is used to highlight software architectural changes and identify incorrect software behavior.

Software interface FMEA analyzes the interface/connection between separate software or hardware elements.

Software detailed FMEA determines the impact of single variable or command failure. Detailed FMEA generally applies to products that do not take advantage of memory protection in the hardware.

Following table shows some of standard failure modes that can be applied to any software functional and interface FMEAs.

FMEA Type	Failure Modes
Functional	Failure to execute
	Incomplete execution
	Execution at an incorrect time
	Errors in the software element’s assigned functioning
Interface	Failure to update a value
	Incomplete update of the value
	Value updates occur at an incorrect time
	Errors in the values or messages

5.0 FMEA Standards and Requirements

Frequently FMEA is a contractual requirement for Military and Defense products. But in general it is not required in the commercial market. However, most companies have realized the value of the FMEA and have adopted it as part of their development process.

Even when FMEA is required contractually, there are no standards for software FMEA. In general software FMEA is vague. Therefore, companies have taken the traditional methods used for hardware and modified them for the software.

The current FMEA standards which are used for hardware are as follows:

- AIAG
- SAE J1739
- SAE ARP5580
- MIL-STD-1629A

These are used for the following types of FMEA :

- Design FMEA (DFMEA)
- Process FMEA (PFMEA)
- Machinery FMEA (MFMEA)
- Functional FMEA
- Software FMEA
- Criticality Analysis

Since there are no formal standards for software reliability, it is acceptable to take the format that is suggested in MIL-STD-1629A and modify it for your own use.

Following are some of the important information that should be captured in your analysis:

- Failure identification
- Failure modes
- Possible failure causes
- State of software
- Detection information
- Who reports the alarm
- Who clears the alarm
- Criticality of the failure
- Failure rate
- How will the failure be reported

6.0 Hardware and Software FMEA Similarities

The approaches to hardware and software FMEA are very similar where the objective is to highlight possible

risks in the design. The exceptions between the two methods are:

Metrics	Hardware	Software
Failure Modes	failure modes are created based on a component or a group of components that make up a function	software failure modes are deriving from a line or lines of software
Failure Cause	an open, short or things like alpha particles and Gamma rays (these failures could cause software failures in Memory)	Programming errors, wrong requirements, incorrect logic or algorithms, bad data and overflow conditions
Failure Rates	Calculation method is based on standards such as MIL-HDBK-217 or SR-332 which is based on component failure rates or physics of failure	Based on lines of code, complexity, CMM level and other parameters. Also, there are models that calculate software failure rates based on development process
Detection Methodology	Using register information	Applying a monitoring software component that uses threads to other software component to check their health
Recovery Method	Retry or replacing the hardware element	Re-start / reboot of the software
Fault Simulation	Removing components or high speed probes	Corrupting the SW

In hardware design FMEA, it is assumed that all other processes are performing to the desired expectation. In other words, we do not look at the failure mode of a digital IC and at the same time bring in the possibility of bad assembly process.

If an engineer is performing the hardware design FMEA, he or she expects to find design or logic related issues and not a bad solder joint.

Similarly in software FMEA, it should be a requirement of the code to be reviewed and then start the FMEA. The

purpose of this process is not to find syntax errors, rather operational and requirement errors as an example.

7.0 Software Failure Modes

As it was stated in section 2, there are several categories of software failures. In this section, we will discuss this in more detail.

For instance, Requirement's failures can be divided into several failure modes

- Incorrect requirements
- Ambiguous requirements
- Conflicting requirements
- Exceptional condition not specified
- Test points or monitors not specified

Let's take another example such as interface failure modes. In this case failure modes are divided into three sub-categories:

1. Functional interfaces
2. Interfaces to hardware
3. Message based interfaces

Each one of these sub-categories is based on a series of distinct and clear failure modes. If we focus on the message based interfaces, we will find the following:

- No message received
- Invalid message received
- Message received out of sequence
- Duplicate message received
- Message not acknowledged
- Message acknowledged out of sequence
- Duplicate acknowledge received

It is essential before any FMEA is started, definition of failure, category of failures and failure mode types are well understood and accepted by all team members. These failure modes will define the roadmap that designers need to follow.

If this step is not addressed, designers will use their own judgment as to what is considered a failure and perform the analysis accordingly. This will create a major obstacle in completion of the FMEA mainly where a number of software components are in stake and more importantly they are operation is inter-related.

One of the keys to a successful FMEA is setting the scope and boundary of the analysis. Identifying and

defining failure modes are the major part of FMEA scope.

8.0 Timing of FMEA and Who should Perform the FMEA

We can perform FMEA at different stages of product development cycle. It should be noted that FMEA details and affectivity are directly related to when it is performed.

Each phase where the FMEA is performed has its own benefits. Generally, the earliest software FMEA can be performed is at the architectural level where different planes (i.e. Control, data or timing) and software components are identified.

The advantage of early FMEA is the ability to define and highlight the system level failure mode categories, expected behavior and critical interfaces. It is natural to expect the accuracy of the FMEA in the concept phase to be limited since most of the detail design has not been thought out.

To prepare an accurate FMEA that reflects the actual design, it is best to have the software designer create and fill in the failure mode information.

9.0 Role of UML Modeling in Software FMEA

UML modeling plays an important role in performing software FMEA, the intent of this section is not to provide a tutorial on UML but introduces the reader to this tool that can show the relationships among different software elements and ease of understanding the design.

The Unified Modeling Language (UML) is a standard language for representing, specifying and documenting software designs of a system. Also is used for business modeling. In general, in case of large and complex systems UML have been proven effective.

The UML uses graphical notations to represent and express the design of software projects. Also, it is an excellent communication tool among engineers to discuss potential designs, possible corner cases and validate the architecture.

There are different UML diagrams that have their own application. Following table summarizes the definitions and applications of each diagram.

Diagram Types	Descriptions	Applications
Use Case	Scenarios describing an interaction between a user and a system	To expose requirements and planning the project
Class	The types of objects in a system and their relationships	The Classes of the system and their relationships to each other
Interaction	The behavior of use cases by describing the way groups of objects interact to complete the task	used when you want to model the behavior of several objects in a use case
State	to describe the behavior of a system	Where it is necessary to understand the behavior of the object through the entire system
Activity	describe the workflow behavior of a system	analyzing a use case by describing what actions need to take place and when they should occur
Physical	Relationship between hardware and software in a system plus software components of a system and how they are related to each other	when development of the system is complete

- How to select failure modes
- Sample failure categories and modes (such as requirements, interfaces, logic and calculation failures)
- Prepare a list of failure modes based on product history
- Create the template or format based on a standard and if not available, an agreed upon format to be used by the team
- Become familiar with the system and software.
- Develop the Reliability Block Diagram (RBD) which is a break down of the system to highlight the HW & SW relationships. It is generally applied more on hardware.
- Perform UML modeling of the software to show the relationships between SW elements.
- Select the Component(s) or subsystem(s) for FMEA .
- Ensure that the selected SW elements have gone through code inspection.
- Set up the team and subject matter experts. This group of engineers is responsible to develop the failure modes and affect of the failures at all levels.
- Select the facilitator (he or she plays a big role in the process). There are a set of distinct qualification for a person to be the facilitator.
- Develop a database to store the FMEA information (off the shelf tools)
- Perform the FMEA
 - Highlight design weaknesses
 - Identify corrective actions as applicable
- Prepare the FMEA report
- Implement the corrective actions

The template below (standard 1629) is tailored for hardware FMEA. As it was mentioned earlier in the paper there are no specific standards for software FMEA. Therefore, it must be modified to the situation at hand.

10.0 FMEA Process

It is a good practice before the start of any FMEA a set of housekeeping steps be followed:

- Define the scope of FMEA
- Collect information such as design specifications
- Create a FMEA guidelines document which as an example describes:
 - Definitions
 - Goals
 - Component definition
 - Interaction definition

FMEA Per 1629 Format									
SEQUENCE NUMBER	ITEM NAME AND FUNCTION	FAILURE MODES	FAILURE EFFECTS	LOCAL EFFECTS	NEXT HIGHER LEVEL EFFECTS	END EFFECTS	SEVERITY CLASSIFICATION	FAILURE DETECTION METHOD	FAILURE ISOLATION
Reference designation or identification number for each failure	Type of hardware or function name	Probable failure modes for each item/function	The consequences of each failure mode on item operation or function	The immediate impact	The effects of the failure as it would be seen at the Next Higher Level	Total effect of the failure has on the operation, function, availability or status of the system	A severity classification category assigned to each failure mode depending upon its effects on system operation	Description of failure detection mechanism	Description of how the failure once detected can be isolated to a single root cause

Following is an example of a SW FMEA based on 1629 standard with modification to its format and list of attributes.

FM ID	Category	Class	Failure Mode	Description	Failure Case	Affected Components	Chance of Failure	Failure Rate (FITs)	Bandwidth loss	Exposure Duration	Traffic Impact	Traffic Weight	Total Bandwidth hours lost
Fabric-1	Protocol	3	Protocol A Switch Failure	Protocol A sends special commands to Logical fabric.	Path not configured for protocol A	Logical fabric	Low	0.01	2304	12	Loss of Traffic	1	276.48
Fabric-2	Interface	1	Activate Line Card failure	Incorrect LC # parameter in message received is not detected	Incorrect parameter, LC Number incorrect.	Program Controller Module	Low	0.01	384	24	Loss of Traffic	1	92.16

FM ID	Local	system	Network	Detection Point & Method	Reporting Mechanism	Recovery Method & Process	Alarm Raised By	Process of Clearing the Alarm	Design Recommendation Summary
Fabric-1	None	Loss of protocol A restoration for multiple connections	No recovery on multiple path	Parameter Validation in Logical fabric	None	retry 3 times	Protocol module	reset protocol module	Design a reporting message
Fabric-2	Program Controller will activate the wrong Line Card. The correct Line Card will not be activated and will not be included in the program.	Insert signal type A on the Line Card that should have been activated	Insert signal type B or cause a mismatch	Via Interrupt	Fabric not capable of activating line card	Reset fabric card	Fabric driver	replace fabric card	none

The following table shows the historical estimated time to complete a FMEA based on product complexity.

Estimated FMEA Completion Time based on Complexity (Hours)				Accuracy
Type	Low	Medium	High	
Functional	40	60	100	Low
Interface	80	120	160	Med.
Detailed	120	160	> 200	High

Summary

This paper described the role of the FMEA and how it can improve design safety and reuse. This technique becomes even more relevant and necessary when companies are challenged by global competition and required to shorten the product development cycle and at the same time reduce cost. The information presented in this paper can be used to build reliability into software and at the same time maintain the goal of reducing the time to market with high degree of confidence that critical, major and even minor failure modes have been identified and addressed (corrected or prioritized).

Following are some of the points that readers can take away from this paper:

- A proactive design analysis that points out the key design weaknesses and help develop the appropriate solutions
- Can be performed at different levels based on project needs
- FMEA provides a knowledge database of all possible failure modes associated with the software
- Should be performed early in the design process to be most effective
- The main source of possible failure modes is the software designer
- A good practice to apply code review before FMEA
- Apply UML approach to identify relationships and interfaces between software modules or components
- FMEA accuracy is directly related to the phase of development and type of analysis
- Commitment of management is essential to the success of FMEA
- Project schedule and budget needs to accommodate the execution of FMEA in the development process
- FMEA has to be taken seriously and subject matter expert's time need to be spent effectively

Nematollah Bidokhti

He is a technical leader at Cisco Systems. His background includes hardware, software and system Reliability engineering, Fault management, System and network modeling. He has contributed and managed reliability activities for military grade, bio-medical, telephony, optical and Data products. He received a BSEE from Florida Atlantic University.