AN ABSTRACT OF THE THESIS OF

Jonathan D. Mueller for the degree of Master of Science in Mechanical Engineering presented on June 5, 2009.

Title: Failure State Identification For Requirements Development During Complex System Design

Abstract approved:

____

Irem Y. Tumer

The increasing level of complexity in systems creates a growing challenge for engineers to design safe and reliable systems. The growing complexity can lead to possible moments when situations occur that were unanticipated or were not known that they could occur by designers and leave the system in an undesirable state. This may happen if system designers were unable to identify the failure state or if they failed to pass on known information to other designers.

This research aims to provide a systematic approach to identifying failure states in complex systems and to improve the connection between the different sides of development of the system by proposing a methodology of investigating the failure states. The methodology identifies potential failure states as a system executes a command and has designers examine them to make recommendations into the severity and potential solutions to the failure state. The information is organized into a single table that is passed over to other system developers and

used in the design of the other sub-systems. The table also serves as a record of the analysis that can be used for reuse or future redesigns.

The benefits of the methodology are examined using the K10 rover developed by NASA as an example. The K10 rover is analyzed to identify its failure states as it executes a command. The identified failure states are analyzed and the information gained is used to classify the failure state according to a ranking scale developed for this research.

Failure State Identification For Requirements Development During Complex System Design

by
Jonathan D. Mueller

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 5 2009
Commencement June 2010

Master of Science thesis of Jonathan D. Mueller presented on June 5, 2009

APPROVED:

Major Professor, representing Mechanical Engineering

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Jonathan D. Mueller, Author

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Irem Tumer for her support, patience and guidance over the past two years. Eric Walkingshaw, Paul Strauss, and Dr. Martin Erwig for their help in my research. Farzaneh Farhangmehr for her help with the development of the software tool. My fellow grad students in the CESD lab for the help and feedback they have provided. The National Science Foundation for supporting my research, NSF CCF-0741584.

It has been a roller coaster two years and I would like to thank everyone that has been there for the ride with me. My friends have been a great source of support, Team Hasselhoff has brought back my competitive edge, third floor Dearborn has given me a good outside perspective, my roommates at the Mat Sci Powerhouse for helping me relax and all the great food, Bombs Away Cafe, Squirrels, Block 15, Brew Station, Interzone and all the other fine establishments of Corvallis for a break away from everything. I would also like to acknowledge my friends from Minnesota for their continuing support and encouragement and Gustavus Adolphus College for preparing me for this opportunity.

Most of all I would like to thank my family. I would not have made it this far if it was not for all that you have done for me and I greatly appreciate that. I would like to dedicate this work to my parents, and I hope you realize that I will always be a Minnesotan at heart! Thank you for everything!

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

FAILURE STATE IDENTIFICATION FOR REQUIREMENTS DEVELOPMENT DURING COMPLEX SYSTEM DESIGN

1 Introduction

The increasing level of complexity in systems creates a growing challenge for engineers to have a complete understanding of the system and the interactions of components and subsystems. Modern systems require several specialties in engineering and design to create systems that consist of several components and subsystems working together to meet the functionality and appeal that customers demand. The growing complexity can lead to possible moments when situations occur that were unanticipated or were not known that they could occur by designers and leave the system in an undesirable state. This may happen if the designers were unable to identify these potential failures or if they failed to pass on known information to other parties involved in the design.

A complex system refers to a system that is composed of interconnected elements that function together as a whole to perform operations that would not be able to do by the elements alone. A complex system can be something living like a beehive or the human body, something electro-mechanical like a car or computer, or anything with different components working together to accomplish a goal together. For the system to be successful, all sides of the system need to be aware of what it takes for each element of the system to be successful as it performs an operation. This research aims to improve the connection between different sides of development of a complex system by proposing a methodology of systematically identifying potential failure states for the system as it completes a process and analyzing them to increase awareness for all sides of development. Designers of the system examine the potential failures to make recommendations into the severity and potential solutions to the failure state. The analyzed failure state information is passed over to the system developers and used in the design of the system. The analysis serves as a record for the system that can be used for reuse or future redesigns.

The benefits of the research are examined using the K10 Rover, a software driven hardware system, as an example complex system. A software driven hardware system is one that utilizes software to run the hardware of the system to achieve the desired functionality. Such electro-mechanical systems include aspects of mechanical engineering, controls engineering, electronic engineering, computer engineering and systems engineering.

1.1 Research Goals

To design more successful systems, it is necessary for system designers to have a better understanding of the different sides of development and to better communicate requirements. To accomplish these goals, this research aims to make a bridge between system designers by identifying potential failures of the system and communicating them to developers that need the information. This will lead to a clearer and better understanding between sides of development in a complex system.

The four goals of

- •Bridging the sides of development for a complex system
- •Identifying requirements for the system as it completes a process and properly managing them
- •Identifying risk for the system
- •Creating a record of the system to be used in design reuse and analysis

are further explored.

1.2 Bridging Sides of Development in Complex Systems

Complex systems require many disciplines of study to design and develop. For example, software developers have a much different background and training than a hardware designer would have. Hardware and software designers use different terminology and it can be confusing trying to communicate necessary information for the development of the system.

When different parties involved in the development of a system do not have good communication, it can lead to a poorly designed system. An example of this is the Mars Climate Orbiter (MCO) that was designed by NASA. The software and hardware developers each used a different set of units in their designs and calculations. The hardware and software were put together and the mission ended in failure when the system failed due to the miscalculations between the hardware and software [1]. Tools or a methodology to ensure proper communication can lead to more successful systems.

1.3 Requirements Identification and Management of Complex Systems

When requirements are not identified and prepared for or properly managed for a system, it can leave it in an undesirable state and potentially end in failure. An example of this is the Mars Polar Lander (MPL) designed by NASA. The hardware designers identified a requirement for the system as it prepared for landing on a surface. The requirement was not communicated to the software developers and a signal was allowed to propagate that cut off the rockets prematurely during its landing sequence and the mission ended in failure when the MPL crashed into the surface of Mars [2, 3]. If the requirement had been effectively managed and communicated to the software developers, the mission may have been more successful.

1.4 Risk in Complex Systems

Complex systems can have several interconnections between the subsystems and its components. This can quickly become complicated and difficult for system designers to have a complete understanding of what is going on and what can happen. If information for the operation of the system is unknown or unavailable, it poses a risk to the system and its operation. System designers need to be able to identify risk and be able to effectively control it to ensure safe and reliable operation.

The Ford Focus debuted in the United States for the 2000 model year. The car was a sales success but the car was plagued with system failures including wheels falling off, roof pillars that may cause damages during a vehicle rollover, inadvertent deployment of the airbags, engine compartment fires, and engine stalling. These failures have led to it being the most recalled car in recent history [4]. Engineers working on the design need to know how much risk they are taking to avoid embarrassing failures that can leave a poor image for organizations.

1.5 Design Reuse in Complex Systems

The aging of the baby boomer generation is leading to the retirement of many working professionals. With their retirement, they also take years of knowledge about projects and systems that they have worked on. If the information was not properly documented for future use, it could be very costly to the companies and organizations when developing new systems. The problem is perceived as so severe in Japan, that it was named the "2007 Problem" there [5, 6].

Systems are expensive to design and upgrading or using parts of previous systems is a way to help reduce development costs. If interconnections from previous designs with updated features are unknown, this could leave the system in an undesirable state or cause failure. An example of this is the Ariane 5 rocket [7-9]. The Ariane 5 rocket was upgraded from the Ariane 4 rocket by putting in a more powerful engine. When this was done a crucial exception handler in the software that controlled the burn time of the engine was neglected and the mission ended in failure when the rocket exploded as it was launching. A methodology that successfully captures the requirements of operation for a system as it operates is necessary to avoid disasters such as this.

1.6 Contributions of Research

This research aims to improve the design of complex systems by addressing the four goals that were stated earlier. This research utilizes several of the methods that are currently used and improves on areas where methods are lacking. The methodology developed uses a model structure inspired from IDEF0 that investigates processes that systems perform. The model breaks down to an event sequence of the process and investigates the components acting on each event in the sequence and what possible states are a possible outcome to identify desirable states (will not cause harm to the system), and failure states (undesirable states that may cause harm to the system or endanger the system from completing its goals). Functional modeling flows are used to demonstrate the change of functionality and help designers brainstorm how the system performs its functions. The identified failure states are analyzed for the possible harm they may cause to a system and what other sub-system designers need to be aware of to handle the situation. The analyzed failure states are used to determine a reliability ranking for the process that is used to determine reliability of event sequences of the processes a system performs.

The methodology is a systematic analysis of the systems processes to determine possible failures of the system and to communicate the analysis to other sub-system developers to use in their designs. The analysis acts as a design record for future analysis of the design and to be used in future redesigns or reuse of the system.

2 Literature Review

Space missions still have failures, cars still get recalled, and other complex systems continue to have failures. There has been a lot of research into designing complex systems, but it is obvious we still need to make inroads if system failures are still occurring. Failures can be expensive, embarrassing, damaging to an organization and even deadly.

One reason for these failures is the growing complexity of systems that can lead to a disconnect between system developers and ineffective communication or a lack of understanding between them. This research aims to have a single methodology to identify potential failures, give input into the potential failures by system designers, document the analysis and communicate the results. This research is different in the method of identifying the potential failures and how it aims to be a single methodology and accomplish the four goals stated in the introduction.

To be able to accomplish the goals of the research, several areas of research are combined to put together a complete model. Ideas from functional modeling are used to help capture change in functionality and a process modeling hierarchical structure is modified and used as a basis for the structure of the models. These ideas are used in conjunction with event sequence diagrams to form a model of the system that captures event sequencing, change in states, related software inputs and outputs, and change in system flows to identify undesirable states of the system to be analyzed as a failure state.

An overview of what is currently being done to try and accomplish the four goals of this research is performed.

2.1 Bridging Sides of Development in Complex Systems

The design process is the organization and management of people and the information they develop in the evolution of a product [10]. How the design process is executed is crucial to how successful the system will perform. For complex systems it is crucial to have all sides of development of the system work together and have knowledge of the other sub-systems to ensure that they work well together.

Concurrent engineering is an area of study that examines different techniques to develop a more integrated design approach between different sides of development to avoid a "throw it over the wall" disconnected design approach [11]. Techniques such as scheduling [12], project planning [13], group activities [14], clustering of design activities into groups that allow effective organization [15], and brainstorming [16] between all sides of development have been proven to lead to a more integrated design team and increase awareness between sides of development.

Software packages have been developed that help with concurrent design approaches. Software such as IBM's Lotus Notes features scheduling, planning tools, and virtual meetings to help designers coordinate meetings and keep each other up to date on the system design [17, 18]. SysML is a standard language that provides a way to incorporate system information from all sides of development together in one place [19, 20]. SysML features several models and functional, interface, physical, performance, and design constraint categories. Corporations such as 3M use company intranets to create storage spaces that designers involved in the system can access and keep up to date on design decisions and changes from any location.

This research aimed to bridge different sides of development by having sub-system designers compare their sub-system to others to find possible situations where the two sides may not work well together and to identify when the two sides will work well together.

2.2 Requirements Identification and Management of Complex Systems

2.2.1 Requirements Management

Failure states need to be identified and managed properly to ensure that designers are aware of them and can prepare the system to handle them. Requirements management is an area of research that studies managing information and is reviewed as a way for designers to manage information about failure states.

Requirements management is a well-researched field that incorporates many different areas of study. There is three main steps of the requirements management process; elicitation, analysis and specification [21, 22]. The three steps of the requirements management process are ongoing and interchangeable and do not necessarily occur in order.

Elicitation is the process of identifying needs and bridging the disparities among the involved communities for the purpose of defining and distilling requirements to meet the constraints of these [23]. Elicitation follows the basic steps of fact-finding, information gathering, and integration [24, 25].

Requirements analysis is the process of interpreting the customer attributes and deriving explicit requirements that are understood by the engineers and the business side of development. This includes classification, prioritization, and negotiation of customer needs [21]. There are several classification schemes [26, 27] used to help guide the designer in compiling, organizing, and analyzing product design issues.

Requirement specification is the process of taking the product requirements created during the requirements analysis stage and turning them into concrete and precise requirements based on functional knowledge [27]. The House of Quality (HOQ) [28] and the Analytic Hierarchy Process (AHP) [29] can be used to accomplish this goal. The HOQ is a graphical matrix that is part of Quality Function Deployment (QFD) [30] and uses a planning matrix to correlate the customer needs to how the organization is going to meet those needs to ensure that a quality product is developed [31]. AHP is similar but it has the ability to analyze the requirements quantitatively whereas HOQ is generally qualitative.

Even with a well-defined requirements management process, there are still large areas for errors or mistakes to be made that can lead to an incomplete understanding of the system [32]. Designing a system with teams of engineers involves several complexities from technical to psychological factors [33]. These many issues can open the door for problems to exist if they are not well controlled. Several problems have been identified that can exist such as flaws in the safety culture of a group, management and organizational factors and technical deficiencies [34]. Techniques such as concurrent engineering approaches [35] can help mitigate some of these issues, but several mistakes may still slip through.

Systematic and automatic approaches to the requirements process are currently utilized and researched in engineering and software fields. Requirements engineering (RE) is a systematic approach to requirement analysis that is used in software and systems engineering to aide designers and help them avoid mistakes in requirements management [36]. Software Process Improvement (SPI) is an accumulation of models that aide designers in the development of software and it includes models for requirements management [37]. RE and SPI are evolving fields and feature many different tools to aide designers with requirements management. The various tools and models can become confusing or designers may not have the proper education to correctly choose the correct model or use a model appropriately. To help with this situation, several tools have been developed to guide designers through the development process and ensure that they are correctly using the tools to assist in requirements management [38, 39].

Design changes are common during the development of a system and it can be difficult to ensure that requirements that were identified are still being considered in the updated design. Requirements traceability is the ability to follow the life of the requirement in both the forwards and backwards direction and is used to make sure that requirements are properly applied to designs [40].

In this research identifying the failure states for the system is the elicitation of the requirements and work done on analyzing and quantifying them is the analysis and specification of the requirements. Creating a clear method of identifying, recording, and communicating information is done to ensure proper management of identified failure states in this research.

2.2.2 Requirement Quantification

Once failure states are identified, they need to be analyzed so designers know where vulnerabilities of the system are and what they need to know to avoid potential failures [41]. There have been several methods of identifying risk including numbers, colors, terms, and different visualizations.

FMECA quantifies the risk of a failure mode into the RPN discussed earlier. The single number is an easy way to portray information that is easily understood by other people, but it also can be too simplistic and hard to portray all the information needed in a single number. Colors have been demonstrated to be effective by Human Factors researchers [42] to portray information and have been used in rating scales such as the United States Department of Homeland Security

Advisory Chart. The chart goes from green, which indicates a low risk level of a terrorist attack, to red, which indicates a high level of risk. The colors were chosen to reflect a person's natural reaction to the corresponding risk level; cooler colors reflect low levels and hotter colors reflect higher risk levels. The colors have no published corresponding references to what criteria are used to determine each level, which leaves the scale open to the operator's discretion but also leaves it open to criticism for a lack of specificity.

A term or name is another method that has been used to describe the potential risk that a failure state can pose on the system. Computer scientists have used the terms Fatal Errors, Normal Errors, and Warnings to describe possible failures that may occur for the system. Each term has a different meaning of the impact that the error may have on a system and must meet certain criteria to be classified for a certain error level.

Different visualizations of data can also be important when portraying information [43]. Human factors researchers have studied several visualization techniques such as displaying numbers on a scale, as a number, the use of color, brightness, hue, sound and other techniques and how they can be used together to accurately and clearly display information [42].

This work utilizes a color scheme to indicate the level of severity of a failure state. A green, yellow and red scale was chosen due to the level of knowledge that most users have with them. Green generally indicates it is safe to proceed, yellow refers to caution, and red alarms users of a more severe failure.

2.3 Risk in Complex Systems

There are several methods of risk assessment and management used in the development of hardware and software systems and includes Failure Modes and Effects (Criticality) Analysis (FMEA/FMECA), Event Tree Analysis (ETA), Fault Tree Analysis (FTA), and Probabilistic Risk Assessment (PRA) [44, 45].

FMEA/FMECA is used to identify and investigate potential system failures and weaknesses by taking a bottom up approach and looking at the individual components of a system and identifying their failure modes, effects, and causes. The end analysis generates a document that contains all of the information that was investigated. This includes, function, failure mode/failure mechanism, failure cause, failure effects, detection methods and controls, and recommended actions. It also generates a number called the risk priority number (RPN). The severity of failure, probability of failure, and detection ability of failure are ranked on a scale of 1-10 (1 low and 10 high). These three numbers are multiplied together to give the RPN number, which can range anywhere from 0 to 1000. FMEA/FMECA is a good analysis method, but it is highly focused on individual components and can easily miss interactions between components or sub-systems that can lead to failure, which is a goal of this research.

Event Tree Analysis (ETA) begins with an undesirable event and uses forward logic to find all the possible outcomes that are possible from the initiating event by considering all the ways that it can affect the overall system by taking into account whether safety barriers are functional or not. Probabilities can be applied to event trees to calculate the frequency of occurrence of a possible event. ETA is a good visualization tool to see the event chains and barriers of a sequence, but it only investigates one event per tree and can overlook subtle system dependencies.

Fault Tree Analysis (FTA) is a top down approach that starts with a potential undesirable event and works backwards to investigate other failures that are related to the top event to determine all the ways that the accident can happen. FTA draws off of an FMECA and a system block diagram and uses Boolean operators to determine reliability of the top-level event using probabilities of the individual elements and using the logic of the tree. FTA leads to an improved understanding of system characteristics but a drawback is that it is in binary so it is either fail or succeed.

FTAs are used in the design stage and use failure rates that are derived from available sources such as handbooks. Shalev and Tiran have applied condition monitoring methods such as vibration analysis and electric current analysis to FTAs to keep reliability information up to date for a particular system to reflect changes to the system such as maintenance or upgrades [46]. Work has also been done to partly automate the construction of fault trees using system topologies annotated with component-level failure specifications [47].

Probability Risk Assessment (PRA) is another approach to addressing risk in complex systems. It is a systematic and comprehensive approach that addresses risk by investigating the severity of failures and the probability of occurrence.

PRA uses FTA and ETA together to investigate possible failure events and to know what would be affected. Determining the total consequences and multiplying them by their occurrences calculate the total risk.

Reliability block diagrams are a graphical representation of a system that illustrate the logical connection between components and are used for reliability prediction [45]. Reliabilities are strung together in series, parallel, or a combination of both layouts. Reliability block diagrams make it easy to identify weak areas and areas that need improvement but they do not give an in-depth analysis of what failed and why, rather it only tags a reliability rating to a component.

This work is similar to FMEA/FMECA in that it examines potential failures and performs an analysis on them. Through the failure state identification process, interactions similar to those identified in an ETA and FTA are captured. Unlike ETA, the goal of this research is to stop the failure from propagating, so the further effects of propagation are not examined. The failure state classifications are used to give a ranking to the process that can be put in an event sequence with other analyzed processes to create a reliability block diagram to determine risk of a sequence of events.

2.4 Design Reuse of Complex Systems

Design-by-analogy is using ideas from other designs when coming up with a new design. There are two approaches to design-by-analogy, one in the

Artificial Intelligence (AI) [48] and one in mechanical engineering. Both approaches utilize a database of designs to search previous solutions. The mechanical engineering method helps designers find similar functions and the AI approach uses software to come up with solutions to design problems.

McAdams and Wood have developed a methodology within mechanical engineering that builds off of functional design [49]. When beginning the design process, the system to be designed is broken into the functionality that is desired. A design repository of components stored as functions has been created. Designers enter in a functionality that is desired and previous designs matching the functionality desired are presented to give designers ideas of how to meet the functionality that is needed for their system. This is a powerful tool to be used in the early design stages. Other researchers have developed online design repositories to be used by designers.

Goel et. all [50, 51] have done research into design-by-analogy in the AI community and also use a database of previous designs to generate design solutions. Previous designs are saved and classified as structure-behavior-function, function-behavior-structure, or topographical graph models. A design problem is entered and the software searches its database to generate a solution. The systems have proven to be successful in testing.

Design reuse is another term that is used to create new designs from previous designs. Design reuse has been used by companies using CAD to create new designs based off of previous designs and a selection of proven parts that the company keeps in inventory [52]. This has helped them increase the reliability of their products and decrease costs by using a limited number of parts.

This research is meant to help with design reuse and design-by-analogy. The methods examined draw off of databases of stored designs. No database is created in this research, but the end result of the methodology presented in this research would be a good attachment to add to previous designs.

3 Engineering Techniques and Methods

The methodology developed in this research utilizes existing engineering techniques to be able to accomplish the goals stated in the introduction. Other methods have been developed and are also examined.

3.1 Event Sequences

Event sequence diagrams can be used to capture the events that a system must perform to complete a task. An event sequence diagram is a series of blocks put together in order of how they need to occur [53]. Blocks are placed in series to demonstrate the order and if events happen concurrently they are placed in parallel as shown below in Figure 1.

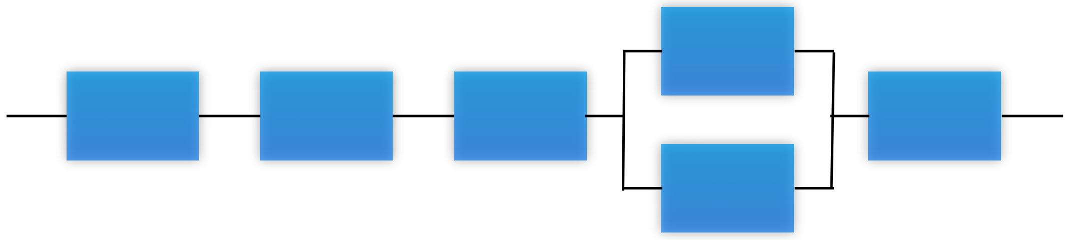


Figure 1: Event Sequence Diagram

Event sequence diagrams are used in this research to show the events that must take place to complete the task of the system to perform.

3.2 Functional Modeling

Functional modeling is a graphical representation of the transformation of energy, material and signal (EMS) flows as they pass through a system. Functional modeling is an effective design tool in the early stages because it places the emphasis on what must be done rather than how it is done [54, 55]. This allows a system to be designed without selecting any specific components or hardware. Functional modeling has also been used for other purposes including modeling failure in a system [56-59] and mapping requirements to the system [60].

The method of functional modeling begins by developing a black box representation of the system. The black box is the highest level of the model and it describes the basic overall function of the system in a verb-noun pair [61]. The black box model shows all of the EMS flows that enter and exit the system.

The abstract black box model can be decomposed into a model of interconnected sub-systems. The model can be further decomposed into components while still retaining the use of EMS flows to describe their function qualitatively. This qualitative model uses a combination of verb-object description for each function.

In this research the event sequence diagrams that are created for a system are connected by the EMS flows used in functional modeling as a way to brainstorm the functionality that is occurring during the process. The functionality occurring during the process is helpful to understand what is going on during the process and what system requirements are needed for the system to successfully complete a process.

3.3 Process Modeling

Process modeling is used in fields ranging from business to engineering as a way to model a process. They range from a simple flow chart to more advanced methods such as IDEF0.

Of these methods, IDEF0 is a structured process modeling methodology that has a development similar to functional modeling [62]. The highest level of the IDEF0 model is the A-0 level, and it consists of a single box that represents the entire process with a verb-noun description. The left side has the inputs into the function of the box. The right side has the outputs, which are the inputs that were changed by the function the box represents. At the top of each box are the controls that manage or act on the functions. Coming in from the bottom are the mechanisms of how the function is performed.

The zero level of the model is split into a group of three to six sub-functions to create the first level. The processes/functions of the first level are individually broken into three to six sub-functions and this represents the second level. The detail of the model increases with the number of levels that are added. This process continues until the level of detail of interest to the designer is reached.

The structure of IDEF0 and a similar numbering system for the model is utilized in this research. The idea of mechanisms acting on the process is modified to be the components that are used for completion of the event. Also, instead of the tools that are used to complete the process, the change in the hardware states will be used.

3.4 Methods

Previous work has been presented in the engineering design literature to combine functional modeling with process modeling, but their work did not identify possible failures that may occur during the process [63, 64].

An extension to their work, the Function Design Framework (FDF) also combined functional modeling with process modeling [65]. The methodology allowed a designer to model a system through any number of events and states. The framework also used a structured modeling hierarchy that allowed for it to be computational. The methodology successfully captured the functionality that occurs at different states well. The proposed methodology in this research focuses more on established hardware and their states rather than the functionality that is occurring.

Function-Based Analysis of Critical Events (FACE) is a methodology that was developed that combines event sequences with functional diagrams and is able to come up with requirements for critical events to be completed successfully [66]. The methodology in this research is similar to the FACE methodology in that they both aim to establish what a system must do to be successful for a sequence of events. This research focuses on component states whereas FACE examines functionality of the system.

3.5 Review

This research aims to systematically identify potential system failure states by analyzing how the system works and the components involved in the process. This is accomplished by combining elements of process modeling, functional modeling, event sequence diagrams, and the states of the components of the system. The methodology identifies a process and splits it into a sequence of events. Functional modeling flows are used through the sequence to understand the changing functionality that occurs during the sequence. Ideas from process modeling are used to develop the model hierarchy.

Together these combine with the states of the components to identify desirable states (will not cause harm to the system), and failure states (undesirable states that may cause harm to the system or endanger the system from completing its goals) for each event in the sequence of events. The failure states are investigated to gain further information into the cause, possible related failures, and solutions to the failure state.

The methodology is similar to other methods that have been developed using similar engineering techniques, but the main difference is that this methodology focuses on component states rather than changes in functionality.

In this research, a process is the term used to define the highest level of the model. A process can be broken down into events, which are sub-processes that when combined create the higher process. Processes and events represent a single or combination of functions.

4 Methodology

4.1 Nomenclature

A few terms need to be clarified before the methodology can be presented. A *command* is something that can be asked of the system to perform a function to complete a task. Hence, a *task* is the system completing the command. A task can be broken up into *events* to complete the task; all of the events together form the task.

4.2 Introduction

This research introduces a methodology that utilizes a unique combination of functional modeling, process modeling with component states to identify failure states for a complex system as it completes a process. The methodology expands to analyze the identified failure states and to classify them based on potential risks they pose to the system. The end result is the failure states of the system that are ranked using a new system based on the potential harm they may cause to the system, possible failures that may result from the failure state, and suggestions from the designers to the other system developers of possible solutions to the failure state. The analyzed failure states are used to rank the process, which can be used in a reliability block diagram with other ranked processes to determine reliability. The clear display of information will help system designers have a better understanding and help make a clear bridge between the two sub-systems of the overall system.

4.3 K10 Rover Introduction

The K10 Rover is a software-driven hardware system that was developed to be a fully autonomous rover to assist astronauts by performing human-paced tasks[67, 68]. The rovers feature several hardware components for taking data and a full drive chassis that allows it to move across the terrain. There is an onboard computer that executes the commands and has the rover complete the tasks that need to be performed. The software controls the movement and the data acquisition for the rover.

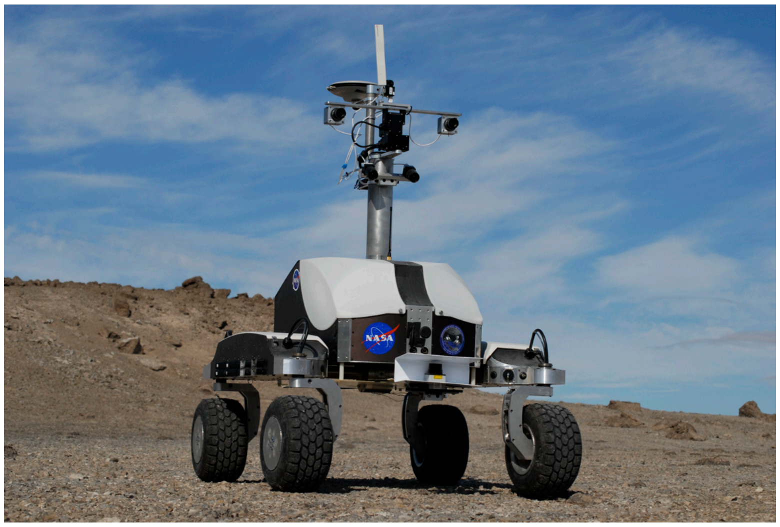


Figure 2: K10 Rover

The K10 will be analyzed from the hardware perspective to identify potential failure states the software side of development needs to be aware of. The system will be examined to identify failure states as the K10 executes a command and completes a process. The identified failure states are analyzed and the information gained is used to classify the failure state according to a ranking scale developed for this research. The information gained from the analysis and the classifications are intended to effectively communicate information needed for successful operation of the K10 Rover to the software developers.

4.4 K10 Example

The benefits of the methodology are illustrated and the steps demonstrated using the K10 Rover at NASA as an example. As mentioned before, the K10 is a good example of a complex system that will be used to analyze the hardware of the system to notify the software developers as the K10 performs a task. An example of a task that needs to be performed is to move the rover to a certain point. This task is crucial for the rover to be able to perform the other tasks that it is able to do. For the rover to be able to move to a certain point, several hardware items need to be utilized and need to work correctly. After the rover moves, it verifies its position by comparing the GPS location with the values recorded by the wheel encoders.

The motivating example occurred during testing of the K10 rover in the summer of 2008, the K10 fell into a failure state that caused the rover to fail and required human interaction to fix. The rover was designed to be fully autonomous so this was a major failure for the system. The failure state occurred when the rover executed a command to move to a waypoint. When it verified its position the GPS distance value did not match the wheel encoder distance values and sent the rover into a failure state.

The steps of the methodology are discussed and are applied to an example of how the task of Move Rover can be analyzed. Failure states are identified and investigated to produce a full record of the failure state.

The example presented is produced from documentation on the K10 Rover [67, 68] and is done as a basic example to show how the methodology can be applied. Assumptions are made about the hardware and how they operate and the material is presented as if it was coming from the hardware engineer for the K10 rover. A more detailed analysis of the system may be necessary to capture all the information.

4.5 Approach

The methodology is performed by completing the following steps:

1) Identify the process to analyze

2) Identify the components in a system, the states that they can be in, the related inputs and outputs from other subsystems, and combine all the information to complete a state table

3) Create a black box model for the task the system is to complete and identify the Energy, Material and Signal (EMS) flows entering and exiting the system on the horizontal plane, the components used in the task, and the states of the components for successful completion of the task in the vertical plane of the diagram

4) Break the black box model into a sequence of events, complete the EMS flows through the sequence, identify the components used in each event and the component states for successful completion of each event

5) Analyze each event to create sub-events and complete the EMS flows, identify components and their states to perform the sub-event

6) Identify the failure states of the system based on the analysis

7) Identify possible causes of why the failure state occurred

8) Identify possible component failures that may occur from the failure state if it is not dealt with appropriately

9) Suggest possible solutions to handle the failure state in an appropriate manner

10) Classify the failure states based on the criteria presented in this paper

11) Consolidate the information into a single table

12) Continue analysis for every failure state and rank the overall process

4.6 Methodology

Step 1: Identify the Process to Be Analyzed

Methodology

To begin the methodology, it is necessary to identify the process of the complex system designers are analyzing that they want to further investigate. The process should be selected so that it can be split into sub-events in later events and represents the highest level of the model, the zero level that is developed by the methodology. The process that is chosen for investigation is illustrated in the model by being written as a verb-noun combination and enclosed in a box as demonstrated in Figure 3 below.

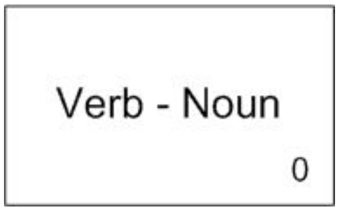


Figure 3: Highest Level of the Model.

Example

The K10 rover performs several actions such as operating ground penetrating radar, moving around, extracting underground soil samples, and taking pictures. The motivating example occurred while the K10 was attempting to move to a target point and that is the process that will be investigated in this example. The highest level of our model is shown below in Figure 4.

Figure 4: Process to be Analyzed for the K10 Example.

Step 2: Create State Table

Methodology

The next step is to come up with a state table that lists all of the components of the system and the possible states that they can be in. Each of the major components of the sub-system of the system that is being analyzed is listed in the first column in the table. It is up to the design team to decide what level of detail they want to go into with their design and the requirements. The designers can be specific by listing every part that goes into the design, or just choosing major components of the system to be more general. The next column in the table lists all of the possible states that each component may be in such as on, off, busy, occupied, non-operational, etc.

The final two columns in the state table are the possible inputs and responses from another sub-system that interacts with sub-system that is being analyzed. The inputs are all of the possibilities that the other sub-system would instruct the component to do. For example, an electric thermometer may be asked to sample temperature by the control software of a system or a worker on an assembly line could be instructed to attach a part by a control board. The final column is the set of possible responses that the component may have to the input it was given by the other sub-system. An example would be the electric thermometer sending the temperature value, sending an error message or maybe no response. Not every component will have connections with other sub-systems and the columns should remain blank.

Example

The K10 Rover features several hardware components that are necessary to complete the process of Move Rover. The state table with the hardware components used in the task Move Rover is shown in Table 1. The hardware components that are used in this task are the GPS, wheels, wheel encoders (alignment and drive), the wheel motors (alignment and drive), and an antenna. These hardware components are listed in the hardware table as well as the possible hardware states for each. Furthermore, the possible software commands and responses of each are listed since we are analyzing the hardware and software interactions.

Table 1: State Table of K10 Rover Filled with Components Used in Example.

Hardware Component	Hardware States	Software Input	Software Response
Wheel(s)	Functional Stuck Spinning Malfunctioning		
GPS	On – Functioning On - Malfunctioning Off	Check GPS	Coordinates Bad Information No Response
Wheel Position Encoder	Functioning Malfunctioning Off	Check Encoder	Position Bad Information No Response
Wheel Distance Encoder	Functioning Malfunctioning Off	Check Encoder	Distance Bad Information No Response
Wheel Position Motor	Functioning Malfunctioning	Turn Wheel	
Wheel Distance Motor	Functioning Malfunctioning	Turn Wheel	

An example of filling in the table for a component is to examine the GPS for the K10 rover. The K10 features a Novatel OEM4 GPS unit. The table can be filled in by examining the user guide of the GPS to find the possible states of the unit and to know what the possible responses are that the unit can provide. Some information may still need to be provided by the designer's knowledge if the user guide is incomplete.

If a hardware item does not exist in the table that is used during the task, it must be added to the table. The bottom of the box shows the hardware states that are possible for the completed task. Some states of the hardware would prevent the system from successfully completing the task, so it is important to show the states the hardware should be in at the completion of the task.

Step 3: Create Black Box Model

Methodology

The black box model is the highest level in the model and represents the zero level that was identified in the first step of the methodology. The black box model adds the components that were identified in the state table that work on the process and the Energy, Material, and Signal (EMS) flows from functional modeling as shown in Figure 5.

All the components that work on the process that were identified in the second step of the methodology are listed on the top of the box for the zero level. The components are also listed on the bottom of the box accompanied by what states are desirable for them to be in. The left side of the box shows the EMS flows that enter the process and the right side of the box shows the EMS flows that exit the process.

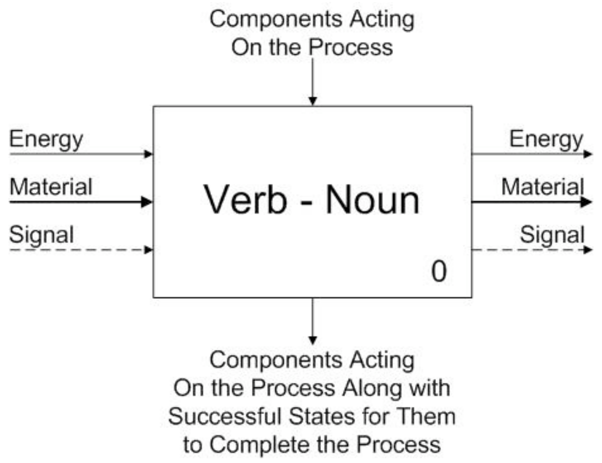

Components Acting On the Process
Energy
Material
Signal
Verb - Noun
0
Energy
Material
Signal
Components Acting On the Process Along with Successful States for Them to Complete the Process

Figure 5: Black Box Model Layout.

Example

To create the black box model for the K10 rover example, all of the hardware that is needed to complete this particular task needs to be considered from the state table created in the first step. The hardware is shown entering the top of the black box model in Figure 6.

The energy, material, and signal (EMS) that are used in functional modeling are listed as electricity, waypoint coordinates (destination) entering the model from the left, and dissipated torque and task status exiting on the right.

The final part of the black box model is to identify what state the hardware components need to be in for the rover to have successfully completed its task to move. The hardware states of 'functional' for the motors, the encoder, the GPS and the wheel are necessary for the successful completion of the task and are shown exiting the bottom of the diagram. The completed black box model is shown below in Figure 6.

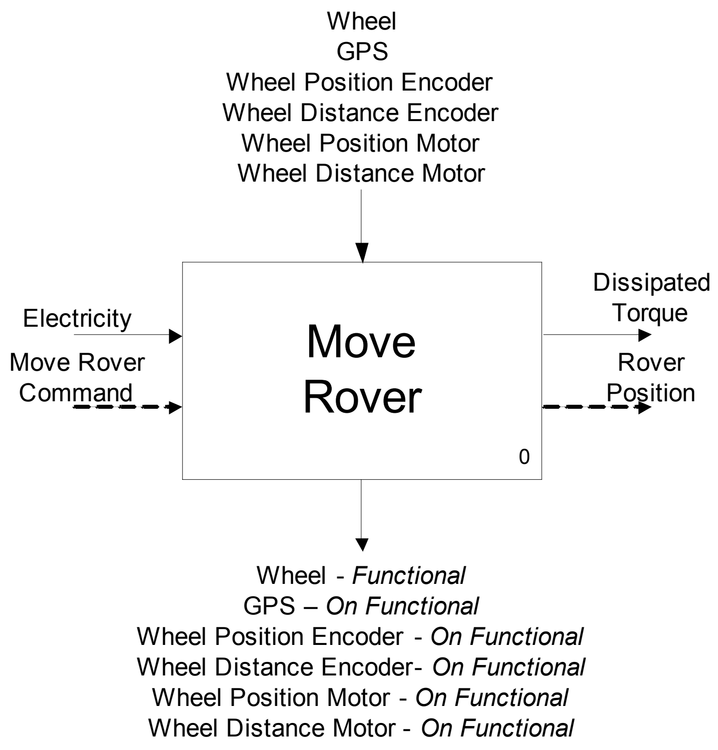


Figure 6: Black Box Model for the Move Rover Task.

Step 4: Event Sequence

Methodology

The next step is to break the black box model into a sequence of lower level events. This sequence of events is the first level, where the events are numbered in the order they are in the sequence. At the first level, the number of events can be as many as the designer feels are necessary, but should be general enough that they will be able to be further split up into lower levels. At this level, events should not occur in parallel; in cases where events are in parallel they should be combined and split up at the next level. Figure 7 shows the numbering sequence and layout of the model at this level.

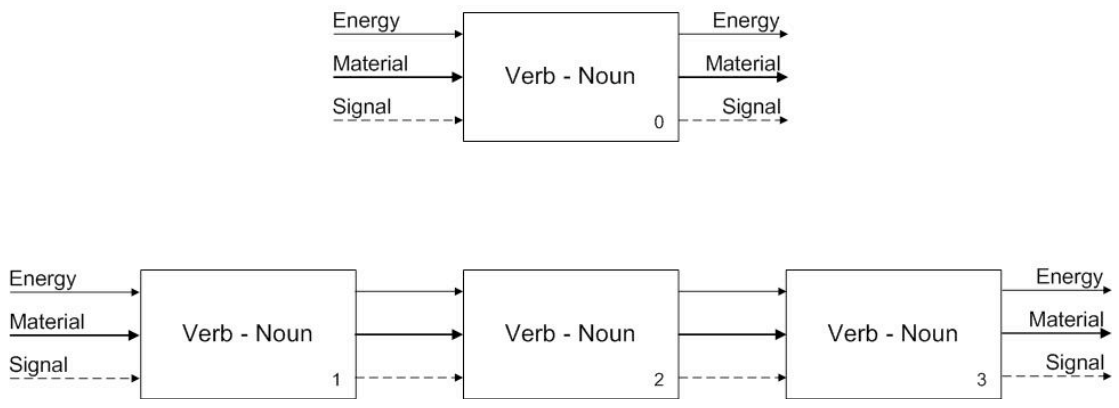


Figure 7: Layout of the Model at the First Level.

The EMS flows enter the first block in the sequence of events and are continued throughout the sequence. The flows follow basic functional modeling techniques and are part of this model to help demonstrate the change in functionality that occurs in the sequence of events.

On top of each box the components that are used to complete the functionality or task of each box is listed. The bottom of each box is the components states that are needed for the task or functionality to be completed and is drawn out at the bottom and connected to the top of the next box in the sequence.

Example

The black box model for the K10 rover example is broken up into the sequence of events shown in Figure 8. These events represent the first level in the diagram and were left general enough so that they could be further broken down into a second level. The event sequence shows that to complete the task Move Rover, the rover must go through the events of Request Waypoint, Receive Information, Move Rover, and Check Position.

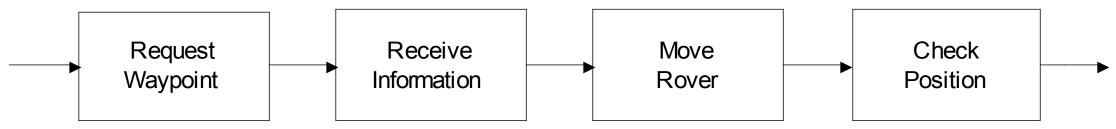


Figure 8: Event Sequence for the Command Move Rover.

The EMS flows that entered the black box model enter the first event of Request Waypoint, and the EMS flows exiting the black box are shown leaving the final event of Check Position. In between these events, the EMS flows are connected and shown in Figure 9.

The hardware that is used to complete each of the events in the sequence is shown as coming into the top of the event. The hardware and the states necessary for completion of the event are shown at the bottom of the event and are connected to the next event in the sequence. The numbering sequence of the events is shown in the lower left corner for each event. Altogether they represent the first level of the diagram and are shown in Figure 9.

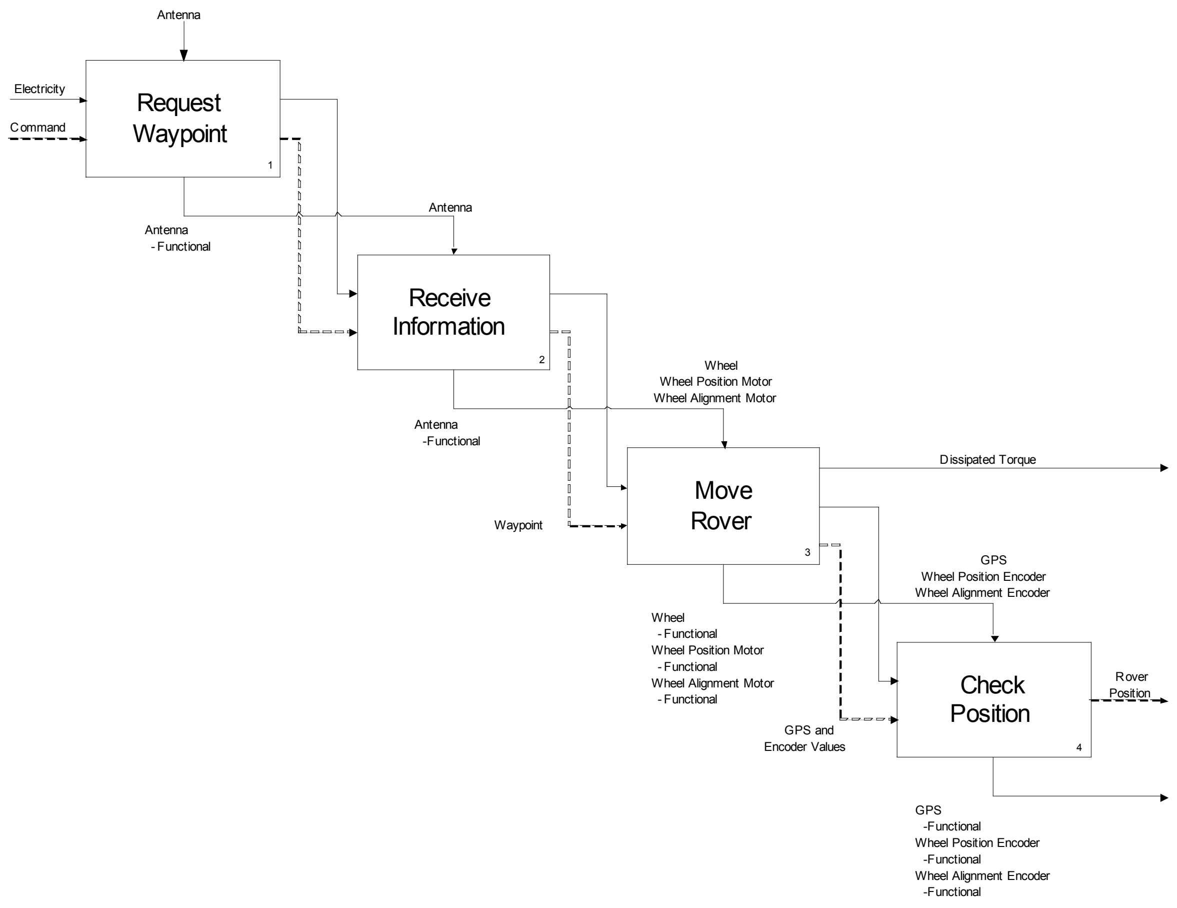


Figure 9: First Level Sequence of Events.

Step 5: Second Level and Beyond

Methodology

Once the first level has been established, each of the events from the first level should be able to be broken down into sub-events to create the second level. The EMS flows that enter the event in the first level enter the first event in the second level for the event. The EMS flows exiting the final event in the second level sequence correspond to the EMS flows exiting from the event in level one. The events can be drawn in parallel, series, or a combination of both.

Similar to the first level, the component acting on each event is listed as entering the top of each event. The components states that would allow the event to occur are listed along with the hardware because some components states would not allow the events to occur. If there are sub-system inputs for the event, they are listed as well.

Exiting the bottom of each event are the possible states that the components can be in along with the sub-system responses that are possible. The states that would allow the system to move to the next event are connected to the top of the next event in the sequence and those that would not allow the system to move to the next event or are undesirable are drawn straight down and do not connect to the rest of the diagram. It is up to the designers of the system to choose what states allow the system to carry out the task and the states that would not allow. This allows designers to have a visual understanding of what can occur in the system throughout the event.

The process of breaking up events and creating a new lower level continues until it reaches a level of detail that is sufficient for the design team; the more levels in a diagram, the greater the detail of the events and functionality as well as the hardware and software interactions.

Example

The events of Move Rover and Check Position are examined in this example. For the rover to be able to move, the wheels need to be positioned (angle) and rotated. The event Move Rover is split up into the sub-events of Align Wheel and Rotate Wheel as seen in Figure 10. These two events happen in series with the wheel being aligned first. The EMS flows entering the event Move Rover are shown entering the sub-event Align Wheel. The EMS flows exiting the event Move Rover exit the sub-event Rotate Wheel and EMS flows connect the two events.

For the event of Align Wheel, the wheel and alignment motor are used, as is shown in Figure 10. They are listed along with their hardware state of ‘functional’ as entering the top of the event. The software input of position wheel is also listed for the motor and wheel.

When the wheel has finished its alignment, the motor and wheel can be in any of the possible states listed in the state table. The states that would allow the system to continue problem-free are identified as the wheel state of ‘functional’ and the ‘functional’ state of the motor. These two states are connected to the next event in the sequence. The wheel states of stuck, slipping, or failure and the motor states of failure are undesirable and are not connected to the following tasks.

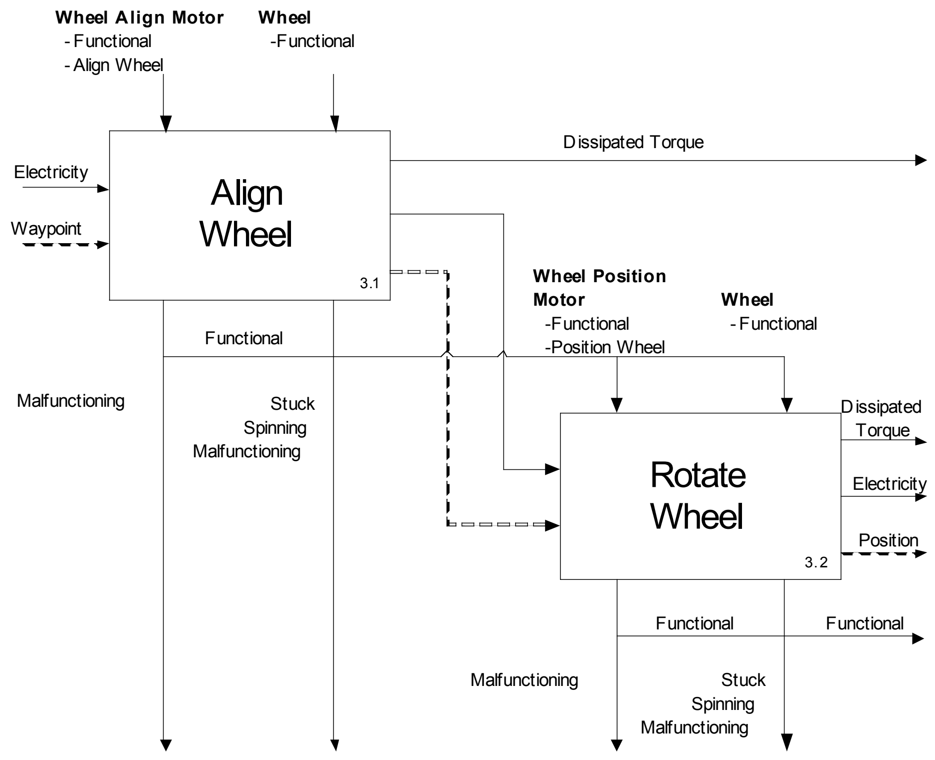


Figure 10: Second Level Diagram for the Event Move Rover.

The event Check Position is for the rover to make sure that it has moved to the correct position. Figure 11 demonstrates how the GPS and the wheel encoders are used to verify the rover's position in this example.

The GPS unit needs to be moving to be able to know the position, but it continually updates to the main computer on the rover. Encoders are devices that convert motion into a series of digital pulses. Knowing the angular resolution of the encoder allows a relative or absolute position measurement. There is a wheel encoder to check the angle of the wheel and an encoder that calculates distance the rover has moved.

The Check Position event is broken into the two sub-events of Check GPS and Check Encoder. The Check GPS event reads the GPS value and the Check Encoder event looks at the encoder values and the calculated position from them. These two events occur in parallel and have the same EMS flows entering them, which are the inputs into their parent event of Check Position. Likewise they have the same EMS flows exiting them which are the EMS flows leaving the parent event of Check Position.

The hardware involved in the event of Check GPS is the GPS unit and is shown as the input to the top of the event of Check GPS. The GPS shows that it needs to be in the functional state and the software command of get position is used to get the GPS value. The input to the top of the event Check Encoders is the wheel alignment encoder and the wheel distance encoder. They must be in the functional state and the software command of check encoders is used to get the values.

For the outputs of the hardware, the states that would allow it to continue to the next task are that the GPS and encoders be on and that they generate the coordinates of the rover. The states of the GPS and Encoders as off, malfunctioning, no response, and bad response would not allow the system to move onto the next task. There is also the situation that the GPS and the encoders both send valid data, but they do not match. This is another situation that would not be allowed to move on to the next event.

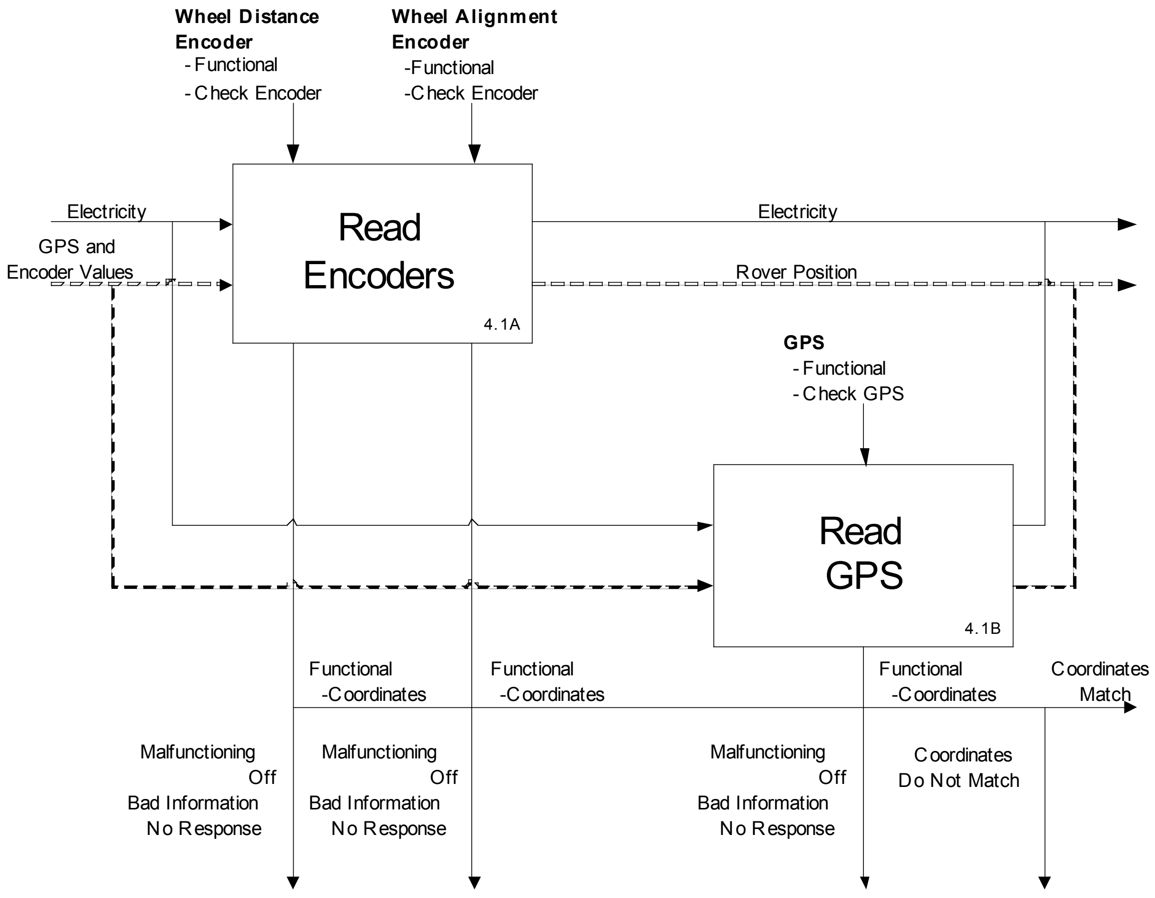


Figure 11: Second Level Diagram for the Event Check Status.

This process can be continued for other tasks for the event of Move Rover and can be continued for the tasks that have already been analyzed. The detail of the model increases with the number of levels that are added.

Step 6: Identify failure states for the system.

Methodology

The failure states of a system are identified as the flows that are undesirable and exit the diagram to the bottom. Every failure is a separate state and should be listed in the format of Table 2.

Table 2: Format of failure states

The system will be in a failure state if:
•Hardware component A is in this state
•The data from component B is not correct
•etc.

Example

From Figures 10 and 11 above, several failure states have been identified and are shown exiting the diagram at the bottom. They are consolidated in Table 3.

Table 3: Failure States Identified for the K10 Rover to Move.

The K10 Rover will not successfully move if:
•The wheel align motor is malfunctioning
•The wheel is stuck
•The wheel is spinning
•The wheel is malfunctioning
•The wheel position motor is malfunctioning
The K10 Rover will not successfully verify its position if:
•The wheel distance encoder is malfunctioning
•The wheel distance encoder is off
•The wheel distance encoder sends bad information
•The wheel distance encoder sends no information
•The wheel alignment encoder is malfunctioning
•The wheel alignment encoder is off
•The wheel alignment encoder sends no information
•The wheel alignment encoder is malfunctioning
•The GPS is malfunctioning
•The GPS is off
•The GPS sends no information
•The GPS is malfunctioning
•The GPS coordinates do not match the wheel encoder values

The failure that occurred during testing in the summer of 2008 for the K10 was caused by the last identified failure state, the GPS coordinates do not match the wheel encoder values. For this failure state to occur, it assumes that the computer of the K10 received values from the GPS and the wheel encoders and that they were in the correct data format. This is the failure state that will be further examined in this paper.

Step 7: Identify possible causes of the failure state.

Methodology

To fully understand the failure state it is necessary to know the possible causes of the failure state. This is necessary because you need to know the cause of a problem before you can fix it. Designers should brainstorm all the possible causes of the failure state and write them down.

Example

The hardware engineers for the K10 rover should be familiar with the hardware on the K10 and the possible causes for this particular failure state to occur. Below in Table 4 are some possible reasons for why the GPS and the wheel encoder values may not match.

Table 4: Possible Causes of the Failure State.

The K10 Rover failure state may be caused by:
•The wheel distance encoder is out of calibration
•The wheel alignment encoder is out of calibration
•The GPS is receiving bad information
•Wheel distance encoder failure
•Wheel alignment encoder failure
•GPS failure
•The wheels may have slipped or be slipping

Step 8: Identify possible hardware failures from the failure states.

Methodology

When a failure state for a system occurs, it is possible that other components or systems may be affected, or could be affected if appropriate action is not taken to handle the failure state appropriately. System designers need to be aware of these potential effects if the system is to properly handle the failure state. Engineers of a particular sub-system are the most knowledgeable of how that sub-system works and all of the interdependencies within that sub-system. For this reason, the sub-system engineer's input is very valuable to the success of the overall system design and how the failure state is handled. Sub-system engineers should evaluate all the possible effects that can be caused due to the failure state and write them down for the software engineers to use when designing the software.

Example

The failure state for this example leaves the K10 rover in a failure state where it cannot verify its position. The possible causes of failure show that the equipment may be operating correctly and only need to be recalibrated, or that one of the pieces of hardware may have a failure. The other possibility is that one of the wheels may have slipped or still be slipping. The first scenarios do not have possible hardware failures that could result from a failure of one of the devices or one of them out of calibration. If the wheel is slipping, there is a possibility that the wheel motor could burn out if the wheel is made to turn too hard or for too long. This is something that could be passed on to the software developers.

There is also that possibility that the rover is not in the location that it was meant to be. The K10 is designed to operate in certain terrain and there is the chance that the K10 may have veered off course and ended up in terrain that is unsuitable for the rover. This information should also be passed along to the software developers. Below in Table 5 the possible hardware failures are listed.

Table 5: Possible Hardware Failures that May Result from the Failure State.

The K10 Rover may experience these failures:
•If the wheel is slipping, the wheel motors may burn out if they are run too long or too fast. •The rover may be in unsuitable terrain that could cause damage to the entire system. The rover should verify that it is able to operate before proceeding with any movement.

Step 9: Suggest possible solutions to the failure state.

Methodology

There are generally several solutions to a problem and that is why sub-system engineers should have a say in how the system resolves a failure state situation. Certain solutions may be better than others and after thinking about the cause and possible effects of the failure state, the sub-system engineers are in a good position to make recommendations of how the system should exit the failure state.

Example

A list of possible causes of the failure state and possible hardware on the rover that may be affected by the failure has been identified in the two previous steps. They should be taken into consideration when coming up with possible solutions to the software developers.

One of the possible causes of the failure state is that one of the components is having a failure. It may be possible to do a hardware check on the components. Reliability ratings for the hardware may also be available and maybe the values of one component can be trusted over the other. If one of the components is out of calibration it may be possible to run the device through a recalibration cycle.

Another identified cause of the failure is that one of the wheels slipped or may still be slipping. This could be verified by checking the slip monitors of the wheels. If a wheel is slipping, the rover could free itself by turning the wheels that are not stuck to free the wheel that is slipping. The possible solutions to the failure state are shown below in Table 6.

Table 6: Possible Solutions to the Failure State.

The K10 Rover may recover from the failure state by:
•Verifying if a wheel is stuck by checking the slip monitors.
•If a wheel is stuck, turn the wheels with traction to free the rover.
•Checking the hardware devices to make sure that they are operational.
•If the devices are operational, run encoders through a recalibration cycle and reboot the GPS.
•If problem persists, the reliability rating of component A is higher than component B, so go with the data from component A.

Step 10: Classify the failure states of the system.

Methodology

It is important that system designers know the severity of the failure states and which ones are important for them to be aware of to handle them properly. Some failure states may not have any potential harm to the system whereas others may cause a catastrophic failure. The classification scheme shown below in Table 7 has been developed to classify the failure states.

Table 7: Classification of Failure States.

Green	The failure state is undesirable but it will not cause any harm to the system. Limited or no system interaction is needed
Yellow	The failure state has the potential to cause harm to the system and action is needed to avoid negative effects to the system
Red	The failure state is likely to cause severe harm to the system and action is needed to correct the situation

The three-tier system allows for hardware engineers to notify software developers of the risks of the failure states and allows them to be open ended so that they can be more specific by entering information collected in previous steps. The information from the previous three steps should be utilized when making a decision to what level the failure state should be classified as.

Example

From the previous steps, it is clear that this failure state could occur and may have several causes. Some of the causes have been identified that could pose harm to the system. According to the ranking system, this failure state would be classified as a Yellow ranking because it has the potential to cause harm to the system but it has a small chance of causing severe harm to the system.

Step 11: Consolidate the information into a single table.

Methodology

The data and analysis for the failure state should be consolidated into a single record to create an easy to read and follow tool for software designers to ensure that the software is prepared to handle this potential failure.

Example

A lot of information for the rover has been collected and gained from the analysis of the previous steps. It is necessary to include all of this information when passing on knowledge of the failure state to the software designers. All of the information should be collected and displayed in a single table that is easy to read. This creates a record of the analysis performed on the hardware and can be used for redesigns of the K10 or knowledge that may be of importance when designing a new rover. This is shown in Table 8.

Table 8: Complete Analysis Table for Failure State.

Failure State	Classification	Possible Causes	Hardware Affected	Possible Solutions
-The GPS coordinates do not match the wheel encoder values	Yellow	•The wheel distance encoder is out of calibration •The wheel alignment encoder is out of calibration •The GPS is receiving bad information •Wheel distance encoder failure •Wheel alignment encoder failure •GPS failure •The wheels may have slipped or be slipping	•If the wheel is slipping, the wheel motors may burn out if they are run too long or too fast. •The rover may be in unsuitable terrain that could cause to the entire system. The rover should verify that it is able to operate before proceeding with any movement.	•Verify if a wheel is stuck by checking the slip monitors. •If a wheel is stuck turn the wheels with traction to free the rover. •Check the hardware devices to make sure that they are operational. •If the devices are operational, run encoders through a recalibration cycle and reboot the GPS. •If problem persists, the reliability rating of component A is higher than component B, go with the data from component A.

Step 12: Continue Analysis for Every Failure State and Rank the Overall Process

Methodology

The methodology needs to be continued until every identified failure state for every sub-event of the process is analyzed. The complete rankings of the failure states are used to give the event a color ranking based on the same scale. All other events are analyzed in the same manner and given a color ranking. These rankings are based off of the designers' knowledge of the system and what they believe the ranking should be. When all the events have been given a ranking, a final ranking is given to the overall process based off of the rankings of the sub-events; once again it is up to the designer.

Once the process is ranked, it can be placed in a sequence of processes and designers will be able to evaluate the event order based on its color ranking.

5 Results

A methodology combining process modeling, functional modeling, event sequence diagrams, and component states to identify potential failure states in a complex system was introduced. The goal was for the methodology to be able to accomplish four goals: to bridge development between different sides of development, identify requirements for reliable operation and properly manage them, assess the risk for the system, and document the information to be used in the future. The research accomplished these goals by doing the following:

5.1 Bridging Development in Complex Systems

The research has system developers consider the interactions of subsystems and give insights into the identified potential failure states. The investigation of the failure states structures the thinking of how the failure state may have occurred (possible causes), what can happen if further propagation were to happen (other components affected), and gives insight into possible solutions for the failure state for designers. Overall, the analysis gets designers thinking of system interactions.

5.2 Requirements Identification and Management of Complex Systems

The methodology focuses on identification of potential failure states, but it also does identify desirable states that ensure reliable operation of a system. These desirable states can be used as requirements for the system as it performs processes. The analysis is documented and made available to other system designers to inform them of how the system should operate and what can possibly go wrong.

5.3 Risk in Complex Systems

A color scheme was introduced to assess the risk of a potential failure state. The methodology builds off of the classified failure states to assign a risk level to the event, and finally the process being investigated. The color scheme is a quick and easy way to alert designers of the risk of a failure state, event, and process for the system.

5.4 Design Reuse in Complex Systems

The methodology produces a completed table of information (Table 8) for each failure state. The analyzed failure states are used to assign a risk ranking to events and processes. The analysis is documented and saved for future analysis or reuse.

The methodology proved effective when it was applied to the K10 Rover, a software-driven hardware system. It successfully identified potential failure states for the process of "Move Rover". The failure state of the GPS coordinates and wheel encoder values not matching was further analyzed to gain further information into the cause, possible related failures, and solutions to the failure state. After analysis the failure was given a yellow ranking, indicating that there is chance that the failure state could cause harm to the system. This identified failure state occurred during testing of the rover in the summer of 2008 and required human interaction to fix the autonomous rover. The situation may have been prevented if system developers were able to prepare for the potential failure state.

6 Discussion and Future Work

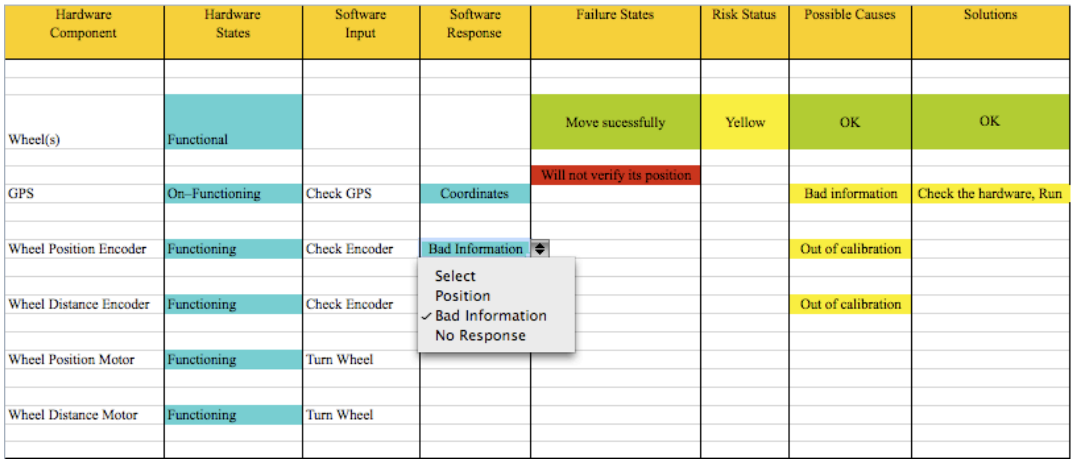
Failures of space missions like the Mars Polar Lander and Mars Climate Orbiter, and operational failures of other complex systems such as the numerous Ford Focus demonstrate that the increase in complexity of systems is making it difficult to design reliable systems. There is a need for more research in adressing the challenge of understanding complex systems. This research addressed the problem by introducing a methodology to identify potential failure states for a complex system as it operates. The methodology expands to have designers investigate the potential failures and to give recommendations into why the failure may occur, what action should be taken to correct the failure state, and assigns a risk ranking to communicate the information to other system developers so they are aware of the possible failures.

The methodology was applied to the K10 rover as an example complex system. Possible failure states were identified for the system as it completed the process of "Move Rover". The methodology successfully identified the failure state that forced human intervention to correct the K10 and recommendations were made for corrective action. If the rover had been sent on a mission to another planet, the failure state may have been the end of the mission, but if the failure state had been identified, the system could have prepared for that particular failure state and taken the appropriate corrective action.

The methodology proved to be successful when applied to the K10 rover example but advancements can be made to improve the effectiveness of the methodology. A tool to aid designers in the methodology and connections with other risk methods and tools would help improve the methodology.

The final step of the methodology can become time consuming and it also needs to be better defined. This step requires designers to perform a full analysis on every event for every process of a system. This could lead to thousands of failure states for a person to analyze which would be impractical. The result of this step of the methodology is not very concrete and relies on the designers' knowledge and what they feel the ranking should be based on the option of choosing between three classifications. A better ranking system, possibly similar to the Risk Priority Number (RPN) in an FMECA would make for more meaningful results. The overall goal is to have each process systematically analyzed for its failure potential and the operation of the system can be modeled as a reliability block diagram and the total risk assessed.

Development of a software tool to help developers overcome these drawbacks has begun as shown in Figure 12. The current version of the software tool performs basic operations of the methodology, but it does not include functional modeling and it currently cannot capture failure states that occur from interactions with other subsystems, such as the one that was investigated in this research (software responses did not match). A more powerful tool that is further developed would greatly enhance the methodology.



Hardware Component	Hardware States	Software Input	Software Response	Failure States	Risk Status	Possible Causes	Solutions
Wheel(s)	Functional			Move sucessfully	Yellow	OK	OK
				Will not verify its position			
GPS	On–Functioning	Check GPS	Coordinates			Bad information	Check the hardware, Run
Wheel Position Encoder	Functioning	Check Encoder	Bad Information			Out of calibration	
Wheel Distance Encoder	Functioning	Check Encoder				Out of calibration	
Wheel Position Motor	Functioning	Turn Wheel					
Wheel Distance Motor	Functioning	Turn Wheel					

Figure 12: Screenshot of the Software Tool.

The goal of the software tool is to automatically identify potential failure states, ease the analysis of failure states, and the ability to save analyzed events to a database. Once all of the events are analyzed designers would be able to construct event sequences with the saved events and the software would automatically calculate the risk. Designers could open the saved processes and events to see the more detailed analysis, or could choose to keep things at a higher level to perform an analysis.

The research has the ability to make a connection with other risk methods and analysis tools to help design more reliable systems. The identified desirable states that would allow the system to continue operation can be used to develop requirements for the operation of the system. These requirements could be input into a SysML requirements diagram to share with other system designers. There is no formal language or specification needed to enter requirements into the diagram, so this could be done now.

The states of failure for the system were identified, but it may be impractical for the system to catch every fault where they were identified in this layout of the diagram. In conjunction with this model, future work will focus on a model that captures when the failures are detected by the system and when they would cause a system level of failure. The methodology presented here captures the possible failures that may occur, which can be complemented by this other model that would demonstrate how the system actually works through them. This would be a good connection into the design of a Prognostics and Health Management (PHM) [69] system to be used with the operation of the system.

The methodology uses functional modeling flows to help system designers to understand the functionality of the event that is being investigated. A connection to the Function-Based Analysis of Critical Events (FACE) methodology that examines the change of functionality during an event sequence for a system would be a strong asset to the research. The two methodologies use a similar approach and a connection of the two models will be investigated in future work.

The research assumes that the system being analyzed operates in ideal conditions at all times. In reality complex systems can operate in varying conditions. A car purchased in Arizona could be driven in the harsh winters of Minnesota and a Mars Rover could encounter a dust storm. A method of accounting for operating conditions on the system and the effect it would have on a system executing processes and a way to incorporate it into the software tool will be investigated.

The methodology was applied to the K10 rover, a software-driven hardware device, and proven effective. Future work includes further investigation into systems that utilize hardware and software, but also to investigate other complex systems, such as ones that feature human interactions. The analysis performed on the K10 consisted of analyzing the hardware and software interactions. Future work will also include investigating systems with more complex interactions.

7 Conclusions

This research presents a methodology to identify failure states for a complex system as it completes a planned task. The failure states are determined by examining the states of the hardware of the system that are necessary for the successful completion of a task. The methodology was applied to a task for NASA’s K10 rover to move to a waypoint and failure states were identified that may pose a threat to the system. One of the failure states was further investigated into possible causes, other hardware that may be affected by the failure, and possible solutions to the failure state. The methodology proved effective in capturing the potential failure states. Future work will explore further reliability integration and the continuation of development of a software tool to aid in analysis.

Bibliography

1. Stephenson, A., *Mars Climate Orbiter: Mishap Investigation Report*. 1999, NASA.
2. Board, J.S.R., *Report on the Loss of the Mars Polar Lander and Deep Space 2 Missions*. 2000, JPL.
3. Young, T., *Mars Program Independent Assessment Team Report*, NASA.
4. *Ford Focus Setting Recall Record*. 2002; Available from: http://www.consumeraffairs.com/news02/ford_focus.html.
5. *The 2007 Year Problem (in Japanese).* Nikkei Monozukuri, 2006.1. 616: p. 48-69.
6. *Blowing Off the 2007 Year Problem.* Nikkei Computer, 2006.7. 657: p. 40-55.
7. Huckle, T. *Collection of Software Bugs*. Available from: http://www5.in.tum.de/huckle/bugse.html.
8. Leveson, N.G., *The Role of Software in Spacecraft Accidents.* AIAA Journal of Spacecraft and Rockets, 2004. 41(4): p. 564-575.
9. Lions, J.L.C., *Ariane 501 Failure: Report by the Inquiry Board*. 1996, European Space Agency.
10. Ullman, D.G., *The Mechanical Design Process*. Third ed. 2003, New York: McGraw-Hill Higher Education.
11. Sage, A.P., Rouse, W.B., *Handbook of Systems Engineering and Management*. 2009: John Wiley and Sons.
12. Belhe, U.a.K., A., *Dynamic Scheduling of Design Activities with Resource Constraints.* IEEE Transactions on Systems, Man, and Cybernetics -- Part A: Systems and Humans, 1997. 27(1).
13. Martin, A., *Project Planning in a Current Engineering Environment.* IEE Conference Publication, 1992(359).
14. Doolen, T.L., Van, A., Worley, J., Farris, J.A. *Designing Kaizen Events for Success*. in *IIE Annual Conference and Expo 2007 - Industrial Engineering's Critical Role in a Flat World*. 2007.
15. Kusiak, A. and K. Park, *Concurrent engineering: decomposition and scheduling of design activities.* International Journal of Production Research, 1990. 28(10): p. 1883 - 1900.
16. Lin, J.Y., *Creativity Engineering for Continuous and Discontinuous Innovation*, in *Portland International Conference on Managment of Engineering and Technology*. 2006: Istanbul, Turkey.
17. Lai, T.a.T., E., *One Organizations's Use of Lotus Notes.* Communications of the ACM, 1997. 40(10).
18. Vandenbosch, B.a.G., M.J., *Lotus Notes and Collaboration.* Journal of Management Information Systems, 1997. 13(3): p. 65-81.

19. Friedenthal, S., Moore, A., Steiner, R., *A Practical Guide to SysML: The Systems Modeling Language*. 2008: Morgan Kaufmann.
20. *OMG SysML 1.0 specification*. November 2008; Available from: http://www.sysml.org/specs.htm.
21. Jiao, J., C.-H. Chen, and C. Kerr, *Customer Requirement Management in Product Development.* Concurrent Engineering Research and Applications, 2006. 14(3): p. 169-172.
22. Zimmerman, M.K., et al., *Making Formal Methods Practical*, in *Digital Aviation Systems Conference*. 2000.
23. Project, S.E.I.R.E., *Requirments Engineering Analysis Workshop Proceedings*. 1991, Software Engineering Institute: Pittsburg, PA.
24. Rzepka, W.E. *A Requirements Engineering Testbed: Concept, Status, and First Results*. in *Twenty-Second Annual Hawaii International Conference on System Sciences*. 1989: IEEE Computer Society.
25. Christel, M.G. and K.C. Kang, *Issues in Requirements Elicitation*. 1992, Carnegie Mellon University.
26. Staufer, L.A. and R.A. Slaughterbeck-Hyde, *The Nature of Constraints and their Effects on Quality and Satisficing.* Design Theory and Methodology, 1989.
27. Fung, R.Y.K., K. Popplewell, and J. Xie, *An Intelligent Hybrid System for Customer Requirements Analysis and Product Attribute Targets Determination.* International Journal of Production Research, 1998. 36(1): p. 13-34.
28. Hauser, J. and D. Clausing, *The House of Quality.* Harvard Business Review, 1988: p. 69-73.
29. Fung, R.Y.K., R. Shouju, and X. Jinxing. *The prioritisation of attributes in customer requirement management*. in *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*. 1996.
30. Thompson, D.M.a.F., M.H., *QFD A Systematic Approach to Product Definition*, in *43rd Annual Quality Congress Transactions*. 1989: Toronto, Canada.
31. Suther, T.W. and A. Sharkey. *Customer requirements research: providing input to quality function deployment*. in *Customer Driven Quality in Product Design, IEE Colloquium on*. 1994.
32. Adler, T.R., *The innovation process: interpreting customer requirements.* Aerospace and Electronic Systems Magazine, IEEE, 1994. 9(6): p. 17-25.
33. Nagamachi, M., *Kansei Engineering*. 1989, Tokyo, Japan: Kaibundo Publisher.
34. Leveson, N., *The Role of Software in Spacecraft Accidents.* AIAA Journal of Spacecraft and Rockets, 2004. 41(4): p. 564-575.
35. Otto, K., Wood, K., *Product Design*. 2001, Upper Saddle River, NJ: Prentice Hall.
36. Sommerville, I. and P. PSawyer, *Requirements Engineering, A Good Practice Guide*. 1997, England: John Willey & Son Ltd.

37. Zahran, S., *Software Process Improvement: Practical Guidelines for Business Success*. 1998, Reading, Massachusetts: Addison Wesley Longman.
38. Jiang, L., A. Eberlein, and B.H. Far, *Combining Requirements Engineering Techniques - Theory and Case Study*, in *12th IEEE International Conference and Workshops on the Engineering of Computer based Systems (ECBS '05)*. 2005. p. 105-112.
39. Jiang, L., et al., *A Methodology for the Selection of Requirments Engineering Techniques.* Software and Systems Modeling, 2008. 7(3): p. 303-328.
40. Macfarlane, I.A. and I. Reilly. *Requirements Traceability in an Integrated Development Environment*. in *Second IEEE International Symposium on Requirements Engineering*. 1995. York, U.K.: IEEE Computer Society Press.
41. Rounds, K.S. and J.S. Cooper, *Development of Product Design Requirements Using Taxonomies of Environmental Issues.* Research in Engineering Design, 2002. 13(2): p. 94-108.
42. Wickens, C.D., et al., *Introduction to Human Factors Engineering (2nd Edition)*. 2003: Prentice-Hall, Inc.
43. Dulac, N., et al. *On the Use of Visualization in Formal Requirements Specification*. in *International Conference on Requirements Engineering*. 2002.
44. Henley, E.J.a.K., H., *Probabilistic Risk Assessment: Reliability Engineering, Design, and Analysis*. 1991: IEEE.
45. Blischke, W.R., Prbhakar Murthy, D.N., *Reliability: Modeling, Prediction, and Optimization*. 2001: Wiley-Interscience.
46. Shalevev, D.M.a.T., J., *Condition-Based Fault Tree Analysis (CBFTA): A New Method For Improved Fault Tree Analysis (FTA), Reliability and Safety Calculations.* Reliability Engineering and System Safety, 2007. 92(9): p. 1231-1241.
47. Walker, M., Bottaci, L., and Papadopoulos, Y., *Compositional Temporal Fault Tree Analysis*, in *Lecture Notes in Computer Science*. 2007, Springer: Berlin/Heidelberg. p. 106-119.
48. Gomes, P., et al., *The importance of retrieval in creative design analogies.* Knowledge-Based Systems, 2006. 19(7): p. 480-488.
49. McAdams, D.A. and K.L. Wood, *A Quantitative Similarity Metric for Design-by-Analogy.* Journal of Mechanical Design, 2002. 124(2): p. 173-182.
50. Goel, A.K., *Design, analogy, and creativity.* IEEE Expert, 1997. 12(3): p. 62-70.
51. Goel, A.K. and S.R. Bhatta, *Use of design patterns in analogy-based design.* Advanced Engineering Informatics, 2004. 18(2): p. 85-94.

52. Andrews, P.T.J., T.M.M. Shahin, and S. Sivaloganathan, *Design reuse in a CAD environment -- Four case studies.* Computers & Industrial Engineering, 1999. 37(1-2): p. 105-109.
53. *Data Process Techniques*, International Business Machines.
54. Pahl, G., Beitz, W., Feldhusen, J., Grote, K.H., *Engineering Design: A Systematic Approach*. Vol. 3rd. 2007: Springer Verlag.
55. Ullman, D.G., *The Mechanical Design Process*. 1997, New York, NY: McGraw-Hill.
56. Tumer, I.Y., Stone, R.B., *Mapping Function to Failure Mode During Component Development.* Research in Engineering Design, 2003. 14(1): p. 25-33.
57. Hutcheson, R.S., McAdams, D. A., Stone, R.B. *Function-Based Behavioral Modeling*. in *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*. 2007. Las Vegas, Nevada.
58. Kurtoglu, T., Tumer, I.Y., *A Graph-Based Framework for Early Assessment of Functional Failures in Complex Systems*, in *ASME International Design Engineering and Technical Conference and Computers and Information in Engineering Conferences*. 2007: Las Vegas, Nevada.
59. Stone, R.B., Tumer, I.Y., *The Function-Failure Design Methoc.* Journal of Mechanical Design, 2005. 127(3): p. 397-407.
60. Hirtz, J., Stone, R.B., McAdams, D.A., *A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts.* Research in Engineering Design, 2002. 13(2): p. 65-82.
61. Stone, R.B., Wood, K.L., *Development of a Functional Basis for Design.* Journal of Mechanical Design, 2000. 122(4): p. 359-369.
62. *IDEF Manual*. 2006; Available from: http://www.idef.com.
63. Nagel, R.L., Stone, R.B., McAdams, D.A., *A Process Modeling Methodology for Automation of Manual and Time Dependent Processes*, in *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. 2006: Philadelphia, PA.
64. Nagel, R.L., Stone, R.B., McAdams, D.A., *A Theory for the Development of Conceptual Functional Models for Automation of Manual Processes*, in *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. 2007: Las Vegas, NV.
65. Nagel, R.L., Stone, R.B., Hutcheson, R.S., McAdams, D.A., Donndelinger, J., *Function Design Framework (FDF): Integrated Process and Function Modeling for Complex System Design*, in *International Design Engineering Technical Conference and Computers and Information in Engineering Conference*. 2008: Brooklyn, NY.

66. Hutcheson, R.S., McAdams, D. A., Stone, R.B., Tumer, I.Y., *A Function-Based Methodology for Analyzing Critical Events*, in *International Design Engineering Technical Conference and Computers and Information in Engineering Conference*. 2006: Philadelphia, PA.

67. Bualat, M., Edwards, L., Fong, T., Broxton, M., Flueckiger, L., Lee, S.Y., Park, E., To, V., Utz, H., Verma, V., Kunz, C., MacMahon, M., *Autonomous Robotic Inspection for Lunar Surface Operations*, in *6th International Conference on Field and Service Robotics*. 2007.

68. Stoker, C.R., Gonzales, A., Zavaleta, J.R., *Moon/Mars Underground Mole*, NASA Ames Research Center.

69. Dabney, T., Hernandez, L., Scandura Jr., P.A., Vodicka, R., *Enterprise Health Management Framework - A Holistic Approach for Technology Planning, R & D Collaboration and Transition*, in *International Conference on Prognostics and Health Management*. 2008: Denver, CO.