SYSTEM DESIGN VISUALIZATIONS FOR SYNTHESIZING INTENT SPECIFICATIONS

by

STEPHANIE SHARO CHIESI

B.S., Massachusetts Institute of Technology (1999, 2001)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF SCIENCE IN
AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May, 2004

Signature of Author ______________________________
Department of Aeronautics and Astronautics
May 2004

Certified by ______________________________
Professor Nancy G. Leveson
Department of Aeronautics and Astronautics
Thesis Supervisor

Accepted by ______________________________
Professor Edward Greitzer
H.N. Slater Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

System Design Visualizations for Synthesizing Intent Specifications

by

Stephanie Sharo Chiesi

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the
Degree of Master of Science.

Abstract

Today's aerospace industry is faced not only with the challenges of developing spacecraft and supporting technologies to explore the unknown, but they must do so successfully with tighter budgets and fewer personnel. Mission failure causes publicity that the industry cannot afford in this economy. To maintain project schedules and prevent budget overruns, problems in the spacecraft system design must be found early in the development stages. An approach to using existing system design visualizations to aid system verification and validation in the early design stages is described. These commonly used system design visualizations are used to create and intent specification in a systems engineering development environment known as SpecTRM. The intent specification is executable and analyzable, allowing system design flaws and requirements problems to be determined prior to any hardware or coding development. An example of the utilization of these system design visualizations to create an intent specification is applied to the mobility and positioning system (MAPS) of a robot designed to process thermal tiles on the space shuttle.

Thesis Supervisor: Dr. Nancy G. Leveson
Professor of Aeronautics and Astronautics

Acknowledgements

The first thing I discovered in writing this thesis is that it is not just about the paper I have here, it is about the journey it took to get here. The journey was difficult, and had far more obstacles than I predicted, but I stayed on track and here is the final product. I truly feel that this process has made me stronger, wiser and much more knowledgeable about my own goals and myself. There have been so many people who have helped me along my way towards completing this thesis and my final degree at MIT and I would like to thank those individuals now.

First and foremost, I would like to thank Professor Nancy Leveson, my advisor. You have opened my eyes to part of the aerospace field that I had not considered exploring before, and you have challenged me in more ways than I think you know. You are truly a great role model and mentor for women in engineering, and I feel extremely lucky to have had the opportunity to work with you. Thank you for your guidance and the remarkable example you set to be the best in the field. Thank you also to Professor Earll Murman and Assistant Professor Charles Coleman for your support, advice and friendship. I will always treasure your opinions.

Huge thanks goes to all of my labmates in SERL, as well as Jeffrey from Safeware for their support and help. You have all taught me so much and have helped me finish this work. Thank you especially Katie, Anna and Nik for your amazing support both in work and as friends. I feel so lucky to now know you so well.

Mike, thank you for all that you are and always have been. You have always supported me in all that I do, and have loved me unconditionally. I hope you know that you are the light of my life. I am so happy to know that I'll always be by your side and that wherever we go it will be together. I love you Sunshine!

Thank you Mom, Dad, Scott, Grandma, Aunt Susie, Uncle Steve, Mom #2, Ada, Aunt Donna, Uncle Dom and all of my family – I love you all! Thank you for never laughing (too loud) at the things I set out to do, and always being with me whether I succeeded or failed. I will always value your love and support and will always be with each of you.

Thank you God for giving me the strength and determination to accomplish the challenges that come with the choices I make in life. My last thought is that I would like to dedicate this thesis to the memory of my grandparents, Sandor Sharo, Helen Sharo and Nicholas Rubino. I know that they have always been with me and will always be watching over me.

Table of Contents

List of Figures

Chapter 1
The Problem

The challenges in the aerospace industry today are no longer just to push the scientific barrier and develop new technologies while successfully completing space exploration missions. Dramatic budget cuts and lack of political and public support have forced the aerospace industry to change the type of spacecraft and systems that are designed and built. The aerospace projects that are now best known to the general public are those that fail before even attempting their exploratory goals. To continue to receive funding for aerospace technology research and space exploration, NASA and other industry leaders must produce spacecraft that can complete successful missions.

Investigation board reports from recent aerospace mission failures show that insufficient or not properly documented requirements and specifications are a leading cause for mission failure. Errors in this stage of system development are far cheaper to correct and take less time away from the development schedule, provided they are detected while still in the design process. This chapter outlines the history of these types of system failures and the problems engineers are faced with in the aerospace industry to lead the way for the systems engineering approach discussed in this thesis.

1.1 Complex Systems in the Aerospace Industry

Spacecraft and other aerospace projects are excellent examples of complex systems. Humans and computers work together to design, develop, build and operate these complex systems to achieve system goals. As the systems in the aerospace industry grow in complexity, a change is needed in the approach to system design. Most of the recent aerospace accidents can be attributed to errors made in requirements and specifications during the system design and development phase. In just one year three different NASA missions (Mars Climate Orbiter, Mars Polar Lander, and the two Deep Space 2 micro-probes) failed due to causes that could be traced to insufficiently written and followed requirements and specifications. Clearly, a change in the approach to writing system requirements and specifications is needed if these complex systems are to be successful given the new budget and time constraints imposed. The aerospace industry cannot afford this lack of attention to system level design and integration to cause further mission failures. Three accounts of these types of failures in spacecraft are described below.

1.1.1 Ariane 5

The maiden flight of the Ariane 5 rocket was scheduled for June 4, 1996. The flight ended in disaster approximately 40 seconds into the initial flight sequences when the launcher deviated from its flight path and exploded. The report issued by the accident review board cited the main cause of the accident as complete loss of guidance and attitude information 30 seconds after liftoff. The flight was nominal until 36 seconds after main engine ignition when the backup Inertial Reference System (SRI) failed, immediately followed by failure of the primary SRI. This loss of guidance and attitude information caused the nozzles of the two solid boosters and the Vulcain engine to swivel into the extreme position and the launcher veered off course. The aerodynamic forces on the launcher caused disintegration to occur in the links between the solid booster and the core stage, triggering the self-destruction of the launcher.

The failure of the primary and backup SRIs can be linked to two different problems in the requirements and specifications for the Ariane 5. The first problem is that it was decided that the Ariane 5 was sufficiently similar to the Ariane 4 that the same software could be reused without making any changes. However, sufficiently similar was not similar enough for this software. The preparation sequence for Ariane 5 was very different from Ariane 4, different enough that an alignment function included in the Ariane 4 software to allow an aborted countdown to restart did not apply to the Ariane 5. This function was "maintained for commonality reasons, presumably based on the view that, unless proven necessary, it was not wise to make changes in software that worked well on Ariane 4" [12]. Though the function was maintained, the software was not tested with the Ariane 5 trajectory data that also differed from the Ariane 4 trajectory data. Had this been done the results would have shown that significantly higher horizontal velocity values from the Ariane 5 trajectory caused an error in the alignment function due to an unprotected variable. This error led to an exception that caused the deflection of the solid rocket boosters and high aerodynamic loads, forcing initiation of the self-destruction of the launcher and its explosion. Had the requirements and specifications not excluded the Ariane 5 trajectory data from system design, analysis and testing, it may have been detected before launch that the Ariane 4 software could not be reused without significant changes.

The second, and possibly greater, problem in the requirements and specifications for Ariane 5, was that the supplier of the SRI "was only following the specification given to it, which stipulated that in the event of any detected exception the processor was to be stopped" [12]. Had the SRI unit continued to at least supply the guidance and attitude system with estimates of position, attempts could at least have been made to still salvage the launcher. However, because of this design flaw, the processor was shut down and could not be restarted. In addition, the redundancy designed for the Ariane 5 was to have identical copies of the hardware and software as backups to the primary system. This means that both systems failed in identical manners due to identical design problems that were in software that was unnecessary for Ariane 5 and not even used in Ariane 4.

1.1.2 Mars Climate Orbiter (MCO)

The Mars Climate Orbiter (MCO) was part of the Mars Surveyor Program (MSP) established by NASA in 1994. Its mission was to orbit Mars as the first interplanetary weather satellite and to serve as a communications relay for the Mars Polar Lander (MPL) that was to land a few months after MCO arrived at Mars. MCO and MPL were to follow two largely successful missions in the Mars program, the Mars Pathfinder and the Mars Global Surveyor. The design teams for MCO and MPL had only 26 months to prepare for these missions in order to launch during the next minimum energy Earth-Mars transfer opportunity. To meet these goals the two project teams had to use what they could from the designs of the previously successful missions, despite differences in the spacecraft and a lack of familiarity with the necessary spacecraft navigation information.

MCO launched on December 11, 1998 and was lost sometime during the Mars Orbit Insertion (MOI) on September 23, 1999. The spacecraft signal was lost over 30 seconds earlier than predicted for Mars occultation loss of signal and never appeared again. The investigation team was able to determine from the spacecraft telemetry that MCO's trajectory was nearly 170 kilometers lower than planned at the time of MOI. This resulted in either the destruction of the spacecraft in the atmosphere or re-entry into heliocentric space [18].

As in Ariane 5, the errors that led to this accident can be traced back to problems in the requirements and specifications for MCO and in the system engineering process for the mission. The "root cause" identified by the investigation committee was that the software file used in MCO trajectory modeling was in English units, rather than the metric units that were specified and expected by navigation teams, creating an erroneous trajectory for the spacecraft. The data from this file was used for each Trajectory Correction Maneuver (TCM) to correct the spacecraft trajectory en route to Mars, but in effect compounding the error.

The Project Software Interface Specification (SIS) to be used for MCO required the output of the application code SM_FORCES to be in metric units of Newton-seconds. This code is used as part of the angular momentum management procedure. To keep the spacecraft's reaction wheels unsaturated, thruster firings are used for Angular Momentum Desaturation (AMD). Each time an AMD event occurs, the spacecraft data is sent to the ground, processed by SM_FORCES, and placed in an AMD file. The AMD files are then used to model the forces on the spacecraft from the thruster firings and determine the spacecraft trajectory. While the software on the spacecraft had the correct units, the ground software did not, and it was the ground version of the AMD files used to determine the spacecraft trajectory. This problem was not even noticeable for the first few months of cruise to Mars because the ground software system was not used due to other reported errors.

MCO is a clear example of difficulties with implementing requirements and specifications in complex mission design. Here the requirements existed, but due to miscommunication and lack of specification use, an accident still occurred. The system engineering process for MCO was not sufficient to ensure that all participants followed the supplied requirements and specifications. The process also was lacking in transitional plans from design to operations, as the trajectory problem could have been detected by the operations crew early enough in the cruise phase to perform another TCM to possibly save the mission, if they were properly trained for MCO operations and the differences in this spacecraft design and operations from those of MGS.

1.1.3 Mars Polar Lander (MPL)

The Mars Polar Lander (MPL) was another part of the Mars Surveyor program. Its 90-day mission was to study the surface environment, weather and geology at the landing site. MPL was designed to send its data to Earth through a relay from MCO. Given that MCO had already been lost when MPL was to land, MPL could also communicate either directly to Earth or through the Mars Global Surveyor satellite.

MPL was launched on January 3, 1999 and arrived at Mars after an 11-month cruise on December 3, 1999. Trajectory correction maneuvers were performed for MPL based on recommendations discovered during the MCO accident review. As occurred for MCO, there was a loss of signal from MPL when entering the Martian atmosphere and the spacecraft was not heard from again. Due to a lack of any telemetry data during the entry and landing phase of MPL, the probable cause for spacecraft lost was determined to be premature shutdown of the descent engines by the flight software.

The landing legs of the lander were to deploy from a stowed state to the landing position at an altitude of approximately 1500 meters above the Martian surface. This deployment occurs while the lander is still attached to the parachute. A Hall Effect magnetic sensor on each leg generates a voltage when the leg contacts the surface. When the first landing leg detects the surface a command from the flight software directs the engine to cut-off. Should the sensor in that leg fail, then the second landing leg would trigger the engine cut-off to prevent tipping the lander over. The flight software also was required to protect against a false touchdown signal or failed sensor in the landing legs.

It was known that the sensors on the landing legs generated a false signal when deployed due to the movement of the legs from stowed to deployed. The flight software was to ignore the signals from these events, but the requirements and specifications of these events were not described to the software engineers, and thus were not properly accounted for in the flight software. Once the lander was at an altitude of 40 meters and the sensor data was enabled, the engine shutdown command was followed and the lander would free fall to the surface and impact at a velocity of 50 miles per hour [1].

The errors found in these three accidents could have been prevented had the system designers taken their design and written requirements and specifications for the mission that properly communicated the information to all those involved in the project. Using the requirements and specifications formulated in this way, a formal approach to validating the system could be applied to detect requirement errors, such as those found in these accidents, long before the system ever leaves the design phase. The next chapter defines system engineering and discusses visualizations commonly used in system design. These visualizations are then used to create an intent specification. Intent specifications and SpecTRM are a formal approach to system engineering and will be described in Chapter 3. The next section describes factors in the aerospace industry that are making the development of successful next generation spacecraft a difficult feat.

1.2 Current Problems in the Aerospace Industry

The aerospace industry is not just plagued with the external problems of budget cuts, public opinion and political sway. There are also difficulties within the aerospace industry that need to be addressed to adequately change the system engineering process and reduce the errors seen in the accidents presented earlier in this chapter. The problems presented here add to the complexity of developing a suitable system engineering process for developing successful spacecraft.

1.2.1 Miscommunication Among Designers, User and Operators

The background, experience and focus of the people working together on any given aerospace project are widely varied. This type of multi-disciplinary project team is common in the aerospace industry, particularly in the system development phase. Spacecraft are composed of a variety of different subsystems including attitude determination and control, communications, power, propulsion, structures, thermal and other specialized subsystems depending on the spacecraft payload. Once launched, flight software controls the subsystems and allows the spacecraft to accomplish the mission objectives. The engineers and scientists on these teams are experts in fields varying from mechanical engineering to computer science to geology. These individuals have extremely diverse backgrounds, talents and communication skills and are often not used to presenting their subsystem needs and abilities to others who have no knowledge of their specialty. The terminology and language each of these experts uses to convey subsystem operation is frequently different, even when referring to the same or similar concepts. Modeling and design tools and problem solving approaches also differ, leaving little to no common ground for system level understanding. In addition, users of the system are often not incorporated into the design phase of the mission, forcing an operator or end-user to adapt to the user interface designed even if it is contradictory to previous training or innate sense. These differences and communication problems among team members can lead to misunderstanding or omission of mission critical details in the system. A common communication platform is needed to facilitate understanding among diverse team members and decrease ambiguity in system design. At the same time, this platform needs to be easy to implement without a significant training period to allow all team members easy access to the information. This will be discussed in the following sections.

1.2.2 Usage of Requirements Documentation in the Design Phase

As was described in the Mars Climate Orbiter accident report, even when requirement and specification documentation exists, that does not always mean it is used by those it is most intended to aide. Not only is the use of requirement and specification documentation not enforced among design team members, it often is not provided to subsystem team members actually designing the details of the subsystem components. When the documentation is provided it is frequently in a format that is difficult to understand or use in subsystem design and modeling. Many times the different people working within a subsystem do not get to provide any input into the document, resulting in requirements and specifications written by those who are not most experienced with that subsystem function. System level knowledge, such as this documentation, needs to be captured and recorded in an easily readable format so that it can be disseminated to all team members working on the spacecraft. Rationale behind the requirement and specification decisions also needs to be documented so that design decisions can be understood and reviewed if needed. This also ensures that the information is passed on to the next generation of spacecraft engineers who need to understand the rationale behind why these system level decisions were made.

1.2.3 Lack of Formal Validation of Requirements and System Integration Prior to Build and Test Stages

Formal requirements specifications are frequently not used in complex system design because it is felt that there is too much training involved in learning how to use a formal model. Formal modeling is not traditionally covered in the curriculum of most engineering disciplines, nor is it exposed to new hires when starting in the aerospace industry, or any engineering related field. As many professionals already in the industry believe formal models to be unreadable and difficult to learn due its base in complex mathematics, their use is extremely infrequent, despite the value gained from their utilization. A platform for using formal requirements specification needs to be easy to learn and easily readable by all those team members who need to use the information it contains. The specification needs to be easy to create by any team member so that as changes and developments in the design process are discovered, the specification can be changed if necessary and the system can be analyzed and executed again to see how the change will affect other system components. With a formal requirements specification, this can happen in the design stages when system changes are still easy and inexpensive, rather than during hardware integration when a quick fix approach may lead to a solution that has farther-reaching system problems.

1.3 Summary

The aerospace industry is currently faced with two different types of problems. The current economy and political climate is limiting the funding available to the aerospace industry. This is compounded by the highly publicized failures of recent space missions, taking away public support from space research. The other problems are those within in the aerospace industry itself. A lack of understanding between engineers with different backgrounds can lead to costly mistakes. Poor usage of requirement and specification documentation in the design stages of spacecraft development can also lead to system level problems that are not discovered until much later in the mission development timeline, causing budget and schedule problems and jeopardizing the mission. As the system design process progresses and design changes are made, the rationale behind these changes must be documented for future knowledge and to ensure that the changes are in agreement with the other design requirements and specifications. With the conception of new and more complex missions, a formal validation of requirements and specifications prior to the build and test phase is necessary to ensure mission success throughout the later stages of mission development.

1.4 Thesis Outline

This thesis proposes using existing system design visualizations to create intent specifications for analyzing and executing complex system designs in the development stages. Chapter 2 provides background on Systems Engineering and some commonly used System Design Visualizations. These visualizations are examples from system design and product development texts that show graphical representations of complex systems. The data displayed through these visualizations will then be used to synthesize an intent specification in SpecTRM.

Chapter 3 describes the systems engineering development platform called SpecTRM. SpecTRM is a toolkit that allows users to create intent specifications that can be analyzed and executed to evaluate the design in the development stages. This allows requirement and specification problems to be detected at an earlier stage in the system development process to avoid budget and schedule overruns, and system failures.

Chapter 4 demonstrates how the system design visualizations feed data into an intent specification in SpecTRM with the Mobility And Positioning System (MAPS). MAPS is part of the shuttle Thermal Tile Processing System (TTPS), which is a mobile robot system designed to service tiles on the Space Shuttle. The goal of this system is to save humans from performing the tedious and potentially toxic task that spans the time between shuttle landing and re-launch, lasting up to 4 months. Finally, Chapter 5 contains the conclusions drawn from the research and example case.

Chapter 2
Systems Engineering and System Design Visualizations

This chapter defines System Engineering and reviews some commonly used System Design Visualizations used in system development. Different sources have different definitions for System Engineering and related terms, so a few different views will be presented here. The visualizations presented are examples from systems engineering texts and papers. These visualizations are different manners of presenting information during the design phase of complex systems. Some visualizations are based on the same information, but present the data in a different manner, showing different relationships involved in the system. Utilizing these existing system design visualizations to synthesize an intent specification is then outlined, with an example of such a synthesis presented in Chapter 4.

2.1 Systems Engineering

As complex systems in the aerospace industry become more heavily reliant on software for system operations after launch, a strong systems engineering development process in the design phase becomes increasingly important. Accidents involving computers are usually due to flaws in the software requirements, not coding errors or hardware failures [7]. Incorporating software development into the systems engineering process from the beginning stages of the system design helps engineers recognize how hardware and software must interact in the system, and allows for more complete requirements specification. Documentation of the requirements is also especially important in complex systems where software designers and other engineers may be very unfamiliar with the functions of other hardware and software in the system. Thorough documentation, trade-off analyses and testing increases the quality of requirements specifications by uncovering problems early in the lifecycle and decreasing the cost of correcting these mistakes. Requirements specifications are analyzed before any code is ever written or any hardware implementation is completed, saving time and money if changes are necessary.

A system can be defined as a group of related components that function together to complete a common goal or task. The systems seen in the aerospace industry are becoming increasingly complex. The factors adding to system complexity are the component interactions and interdependencies within the system. These detailed interactions between system components mean that no individual part of the system can be engineered independently of the rest of the system, or the part may not function properly when it is later introduced to the rest of the system. A view of the entire system with individual component operations and interactions must be used in system development in order for the components to function correctly in their system level capacity. This approach to design is called systems engineering.

Many different definitions exist for the term systems engineering, though all contain similar ideas. A Space Systems Engineering course at MIT defines system engineering as “the ensemble of coordinated analyses, simulations, and processes which lead to the design of a technical product which best meets the needs of an identified customer” [14]. This approach tells the “whole story” of a system, including the why, which, what, how, when, and where, to the customer in order to meet the customer’s needs. Here, “why” refers to the requirements that define the customer’s needs and the point of the mission being conducted. “Which” is a trades analysis to compare different system architectures to determine what best meets the requirements and customer needs. “What” is the design that fully describes the system to be built and operated. “How” is the program plan to provide for an organizational structure for the system, allocation of resources, schedule, and information dissemination. “When” is the schedule for system development stages. Lastly, “where” refers to hardware flow details, obtaining components, the facilities for integration, test, and validation, as well as checkout and launch [14].

Another definition for systems engineering comes from the NASA Systems Engineering Handbook, where systems engineering is “a robust approach to the design, creation and operation of systems” [15]. This approach also consists of identifying and quantifying system goals, creating alternative design concepts, performing a trade-off analysis of these designs, selecting and implementing the best design, verifying that the design was properly built and implemented and finally assessing how well the system meets the goals [15]. While the definition of the term is different, the approach contains many of the same ideas as that from the previous definition.

The International Council on Systems Engineering also has a different definition: "Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem. Systems engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation" [4].

Systems engineering has also become recognized as a separate department for studies at MIT. The MIT Engineering Systems Division (ESD) first defines systems architecture as "the process by which standards, rules, system structures and interfaces are created in order to achieve the requirements of a system; trade-off studies may precede the determination of system requirements." [13] This more closely fits the definitions described above for system engineering. However, the ESD characterizes system engineering as "a process for designing systems that begins with requirements, that uses and/or modifies an architecture, accomplishes function and/or physical decomposition, and accounts for the achievement of the requirements by assigning them to entities and maintaining oversight on the design and integration of those entities…Systems architecting creates a system design at a high, abstract level, whereas systems engineering is often associated with refining such a design; by blending the two processes one accomplishes the assignment of functions to physical or abstract entities, and the definition of interactions and interfaces between the entities." [13] This definition better recognizes the fact that not all parts of system engineering are tangible components, but that software components also need to be factored into the development processes.

Systems engineering is multidisciplinary. The complex systems in aerospace are composed of many different subsystems, including everything from human operations to software to electronics and fuel. Engineers taking part in the system development process come from all different disciplines to contribute to the overall system functionality. Systems engineering is also all inclusive of all aspects of the system at hand. It involves subsystem and component trade-off analyses and evaluation of customer need and requirements satisfaction at each stage in the development lifecycle. The systems engineering approach to complex systems development also means that the engineers working on different system components are cognizant of the rest of the system and how other system components function in the overall system. Finally, systems engineering throughout the system development lifecycle reduces costs by creating design alternatives at each development stage so that a problem at one stage in development does not send the entire system development back to square one.

2.2 System Design Visualizations

Visualizations have long been used a way to enhance cognition of information, particularly large and complex amounts of information [2]. Graphics become a common ground for understanding between people whose backgrounds and disciplinary languages differ. The majority of the published work on visualizations for cognition revolves around displaying data and not on systems or development processes. This does not mean that visualizations are not used in system design, but rather how they function in terms of information understanding has not been widely studied. The visualizations that are described in the following sections are taken from texts on system design and system design documents themselves to demonstrate how multi-disciplinary engineering teams use visualizations to enhance system understanding during the development lifecycle.

2.2.1 System Process Flow Diagrams

System process flow diagrams are common at the beginning of system design to determine system components and the basic links between these components. They are block diagrams where the blocks represent different subsystems in the complex system and arrows from one block to another block represent interactions between the subsystems. This visualization is useful in complex systems because it can be made as detailed or as all-encompassing as necessary depending on what part of the design or design lifecycle is being represented. Figure 1 shows a sample system process flow diagram for a spacecraft modeling project at the highest level. The block labeled Spacecraft Model can then be made into a separate system flow diagram of its own to show the interactions of the subsystems it contains. Notice in this version of the diagram there are no arrows linking the subsystem blocks within the Spacecraft Model block, as that level of detail is not part of this particular diagram.

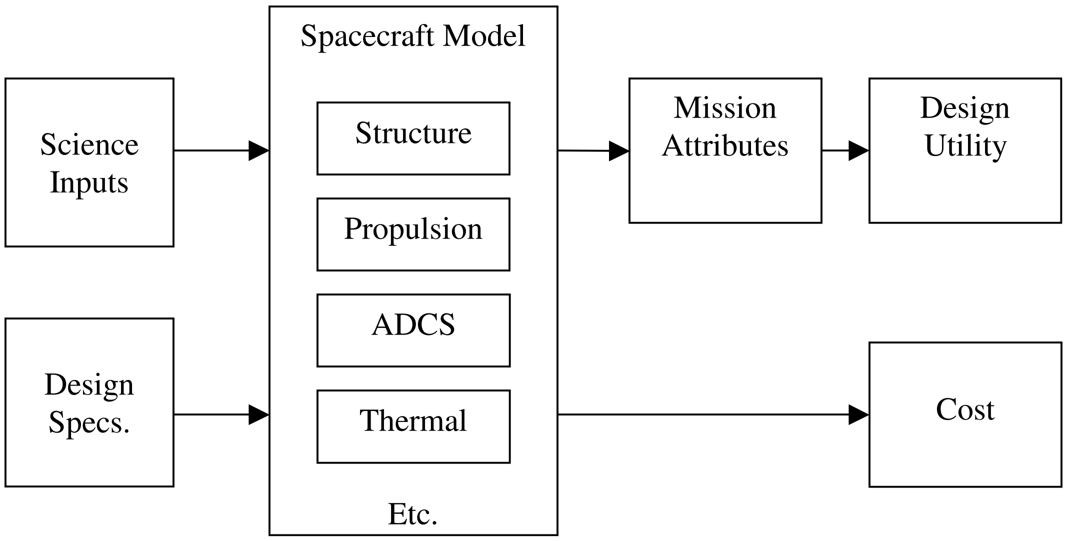
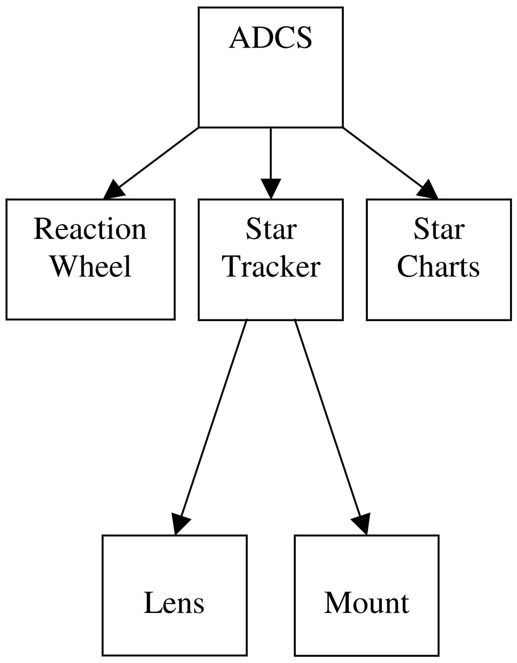


Figure 1. Sample System Process Flow Diagram

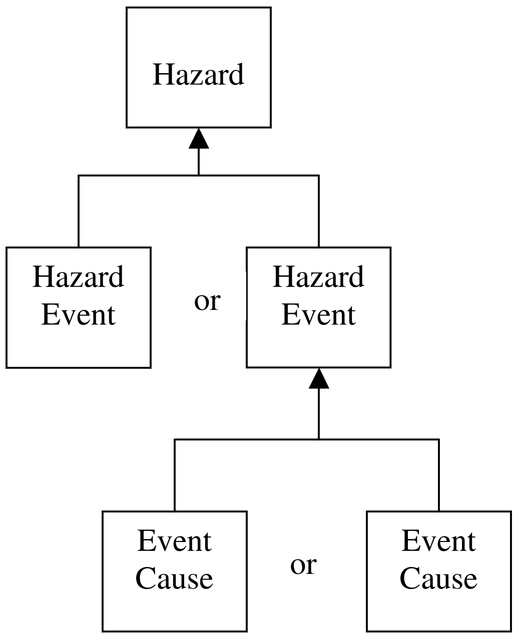
System process flow diagrams are easy to construct and well understood by multi-disciplinary project team members. They also provide a good amount of system level information that can be easily incorporated into an intent specification.

2.2.2 Tree Diagrams

Tree diagrams are another commonly found visualization tool in system development. Tree diagrams are composed of nodes and branches. The nodes can represent different things depending on the function of a given tree, for example if using a tree diagram for a preliminary hazard analysis a node may represent an event that occurred leading to a hazard. A node can also represent a piece of hardware or even a subsystem. The branches in a tree diagram represent unidirectional links to the next node, and often have conditions attached to them. Tree diagrams can be used to converge to one node, or as diverging from one node as shown in Figure 2. This is highly dependent on what information is being conveyed through the diagram. Tree diagrams are useful in identifying parts of a subsystem and the breakdown within a subsystem as is shown in Figure 2 part A. They are also useful in preliminary hazard analysis when brainstorming ways in which it is possible to reach a particular hazardous state as shown in Figure 2 part B.



A



B

Figure 2. Sample Tree Diagrams

Tree diagrams are a very familiar and comfortable visualization for conveying information. They are simple to use and can be expanded or collapsed as necessary for whatever level of design is being addressed. They can also be easily tailored to many different system design needs.

2.2.3 N^2 Diagrams

N^2 Diagrams are useful matrix notation visualizations for showing the interactions of different subsystems and even the actual variables and information needed in the interactions. This diagram is helpful in determining if a reordering or restructuring of the elements in the diagram would help simply parts of the system. Boxes that are checked represent inputs to the subsystem in the column from the subsystem in the row. If the box is checked above the diagonal indicate a linear flow of the system. Boxes checked below the diagonal indicate that information from one subsystem was returned to another and could possibly change the effect on all other subsystem again that required outputs from that subsystem.

Design Vector				X	X	X	X	X	X	X	X
	Science Vector	X	X		X	X	X	X		X	X
		Instr.	X	X		X			X	X	
			Acquistion	X		X		X	X	X	
				Rover Init.	X	X		X	X	X	
					Environ.	X		X			
						Auto. Init.	X	X	X	X	
							Comm. 2	X	X		
								Rover	X	X	
								X	Power	X	
										Auto.	X
											Comm. 3

Figure 3. Sample N^2 Diagram

Sample based on a Mars Rover Design Tool [14]

The sample N^2 diagram shown in Figure 3 shows a system that is mostly linear in function but does have one iterative step, as the box between Rover and Power is checked below the diagonal. This kind of interface information is valuable, particularly when developing the software in the system development phases.

2.2.4 State Transition Diagrams

State transition diagrams are very important in system design. This is a visualization that helps coordinate development of the system states. Depending on the system being designed, this can include operating states, states of individual parts of the subsystems, and states displayed to the user. The executable intent specifications that will be discussed in the next chapter are based on an underlying state machine model, so these diagrams are very useful in creating intent specifications. State transition diagrams show the feasible states of a particular part of a system, and how transitions from one state to another can occur. Figure 4 shows a simplified state transition diagram for a simple spacecraft [5]. The nodes here are the different possible system states. The links between the different states represent conditions for changing from one state to another. In this example there is only one condition on each link, but it is possible in a much more complex system to have multiple conditions that are required to leave a given state.

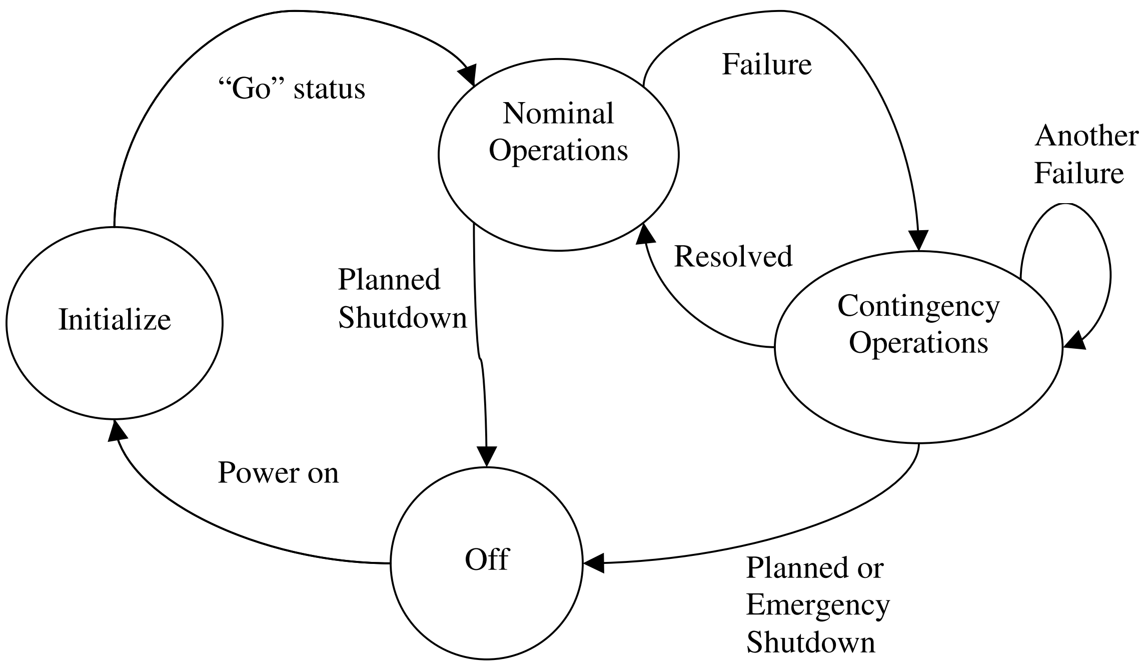


Figure 4. Sample State Transition Diagram

As with the other diagrams shown thus far, this diagram is very easy to create and understand, as well as useful in synthesizing intent specifications.

2.2.5 User and Task Models

User and Task Models are important visualizations in complex system design in helping to identify the potential for mode confusion in a system. The aerospace industry utilizes automation in many systems to cut personnel costs, and also to further research in the automation field. However, the use of automation in complex systems creates different types of problems because of the new type of interaction between the human operator and the automated system. Creating user and task models helps the design team to recognize designs that could potentially lead to mode confusion, where the user or operator is not aware of the mode the system is currently in or how it got there. Figure 5 shows the user and task models created for a flight management system case study [16].

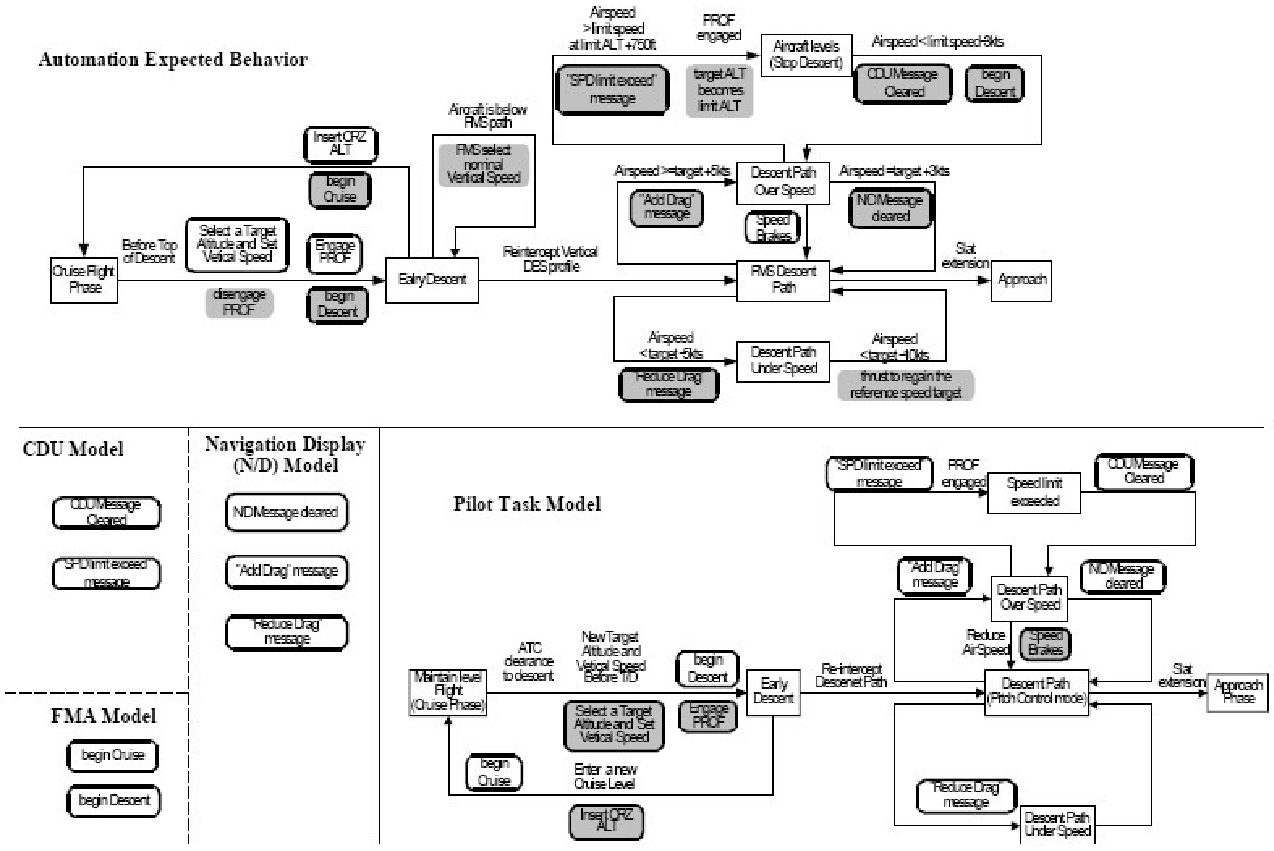


Figure 5. Sample User and Task Model [16]

Information from models such as these is very important in synthesizing intent specifications as accidents caused by mode confusion are often due to a lack of requirements to prevent the confusion from occurring.

2.2.6 Input/Output Diagrams

Input/Output diagrams are blackbox representations of what goes into and out of a subsystem or subsystem component. They can be simple diagrams that only show the inputs and outputs to one given subsystem as shown in Figure 6, or they can show multiple components and the inputs and outputs passed between them. Input and output information is important in system design to ensure that the information that each subsystem needs is generated and passed from other subsystems. Figure 6 shows inputs coming from two different sources and outputs being sent to three different subsystems.

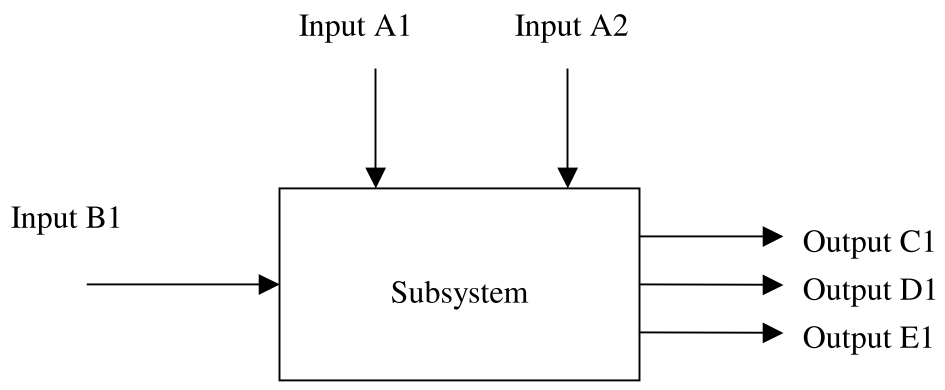


Figure 6. Sample Input/Output Diagram

2.2.7 Key Metric Value Charts

Key Metric Value Charts are tabular methods of representing system design information. Tables can display large amounts of information in an organized and easily understandable format. The types of system information that can be presented in this way includes design limitations, input and output lists separated by subsystems, user defined limits, and much more. Figure 7 shows a sample chart of required inputs and outputs from different spacecraft subsystems.

Subsystem	**Inputs**	**From**	**Outputs**	**To**
Power	Max. Temp. Ranges	Environment	Power Available	Communications
	Power Rqmnt. over time	All Subsystems		
Communications	Power Available	Power	Power Rqmnt. over time	Power
	Data Volume Data Rate Data Storage Rqmnt.	Instruments		
Instruments			Power Rqmnt. over time	Power

Figure 7. Sample Key Metric Value Chart – Inputs/Outputs

This chart would be used in the system design phase to ensure that the different subsystem team members know exactly what information they must be able to provide for other subsystems to consider when designing their own subsystems. For example, the power subsystem would need to know the power each subsystem requires when operating in order to use an appropriately sized power system to accomplish the mission tasks. This information details the links and interfaces between subsystems in another format for multi-disciplinary team members to understand when designing their separate subsystem. Another key metric value chart may display component attributes, as shown in Figure 8 [5].

Subsystem	Component	Weight (kg)	Power (W)
ADCS	Sun Sensor	0.5	0.1
Communications	S-band antenna	0.9	0
	Receiver	1.8	4
	Transmitter	2	4.4
	Diplexer	1.2	0
Power	Solar Arrays	0.04	-
	Batteries	Capacity/35	-

Figure 8. Sample Key Metric Value Chart – Subsystem Component Attributes

Attributes such as these can be used as values to satisfy the input and output information a subsystem needs for system integration. These values are also used when creating executable models for system operations. Data from charts such as these provide the values and variables necessary to give definition to the system design in an intent specification.

2.3 Summary

System design visualizations are valuable tools in systems engineering, particularly in the design phase of the system development lifecycle. Left alone the visualizations do provide useful information to multi-disciplinary team members, but they do no more than just provide information. The data these visualizations contain can be put to much more use in system engineering if they are utilized to synthesize an intent specification. Now the information is not only useful and easy to understand, but it is also executable, and more valuable in determining requirements and specifications needed for a successful complex system. The next chapter describes a systems engineering development platform for creating these intent specifications to enhance the systems development process.

Chapter 3
SpecTRM

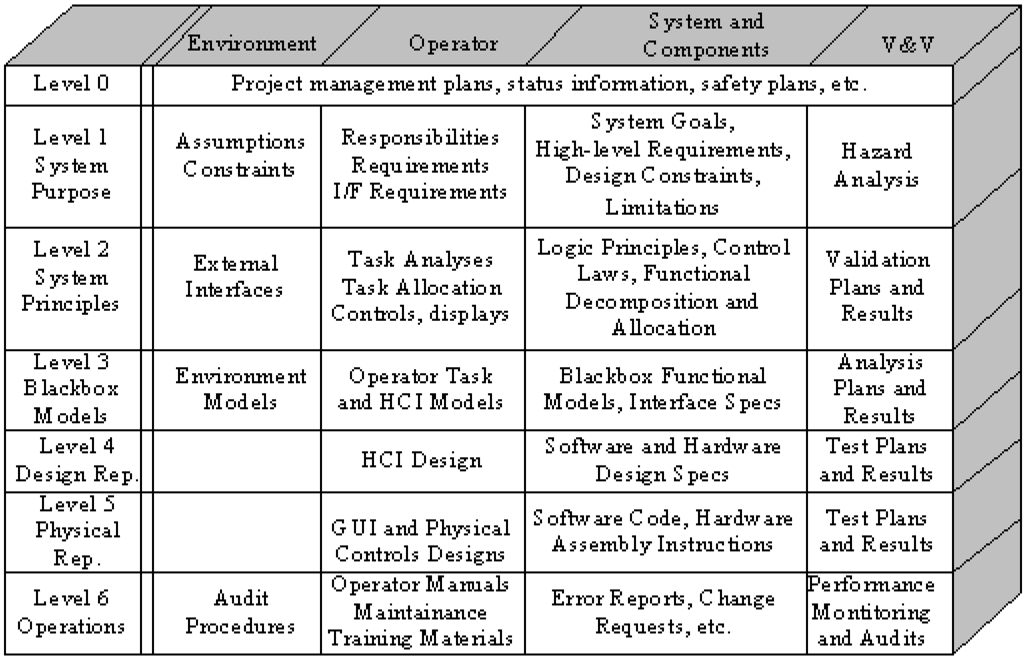
A platform is needed to use the systems design visualizations described in the previous chapter to construct executable system models. This chapter discusses intent specifications and SpecTRM as a platform for this approach to systems engineering. The system design visualizations provide the information for an intent specification that can be built in SpecTRM. SpecTRM is a toolkit for creating intent specifications and performing formal analyses on the model. Finally, a recent system engineering application of SpecTRM is discussed in which separate intent specifications are created for reusable generic spacecraft components in order to utilize SpecTRM as a platform for Component-Based Systems Engineering.

3.1 Intent Specifications and SpecTRM

An intent specification is documentation of the analysis, design, implementation and operation for a system. The structure of an intent specification is based on human problem solving and supports basic system engineering principles. Specifications are used to perform analysis, review software designs, debug, maintain, and reengineer systems as needed. Primarily an intent specification differs from a standard specification in structure. Intent specifications are structured as a set of models designed to describe the system from different viewpoints throughout the entire system development process. The structure facilitates the tracing of system-level requirements and design constraints down into the detailed design and implementation. It assists in the assurance that system properties, such as safety and system security, are incorporated and implemented in the design from the beginning. The structure also reduces the costs of implementing changes and reanalysis when system changes occur. Intent specifications enhance problem-solving performance because they help designers extract the important information necessary for a system without making any modelling assumptions. System designers and engineers are not limited to a specific manner of carrying out the system design tasks in intent specifications, but rather the information needed to accomplish these tasks is recorded and whatever strategy the designer chooses to take in using the information can be taken. No additional specification is involved in this process (assuming that projects produce the usual specifications), but simply a different structuring and linking of the information produced so that specifications are used more often and correctly in the development and evolution process [8].

An intent specification is comprised of seven levels as shown in Figure 9 [8]. It is important to note that the different levels do not represent refinement, as in other more common hierarchical structures. In an intent specification the levels represent a different model of the same system and support a different perspective of the complete system. The model at each level may be described in terms of a different set of attributes or language. Refinement and decomposition occurs within each level of the specification, rather than between levels [8].

The top level, or Level 0, is a project management view of the system and provides insight into the relationship between the plans and project development. Level 1 of an intent specification is the customer view. This level assists system engineers and customers in agreeing on what should be built and whether that has been accomplished. It includes system goals, high-level requirements, design constraints, hazards, environmental assumptions, and system limitations. Level 2 is the System Design Principles level. This is the system engineering level used by engineers to reason about the system in terms of the governing physical principles and laws that apply to the system being designed.



	Environment	Operator	System and Components	V&V
Level 0	Project management plans, status information, safety plans, etc.			
Level 1 System Purpose	Assumptions Constraints	Responsibilities Requirements I/F Requirements	System Goals, High-level Requirements, Design Constraints, Limitations	Hazard Analysis
Level 2 System Principles	External Interfaces	Task Analyses Task Allocation Controls, displays	Logic Principles, Control Laws, Functional Decomposition and Allocation	Validation Plans and Results
Level 3 Blackbox Models	Environment Models	Operator Task and HCI Models	Blackbox Functional Models, Interface Specs	Analysis Plans and Results
Level 4 Design Rep.		HCI Design	Software and Hardware Design Specs	Test Plans and Results
Level 5 Physical Rep.		GUI and Physical Controls Designs	Software Code, Hardware Assembly Instructions	Test Plans and Results
Level 6 Operations	Audit Procedures	Operator Manuals Maintainance Training Materials	Error Reports, Change Requests, etc.	Performance Montitoring and Audits

Figure 9. Intent Specification Hierarchy

The third level, or Blackbox Model level, enhances reasoning about the logical design of the system as a whole and the interactions between system components, including system functional states, without being distracted by implementation issues. Level 3 acts as an unambiguous interface between systems engineering and component engineering to assist in communication and review of subsystem blackbox behavioural requirements and to reason about the combined behaviour of individual components using informal review, formal analysis, and simulation. The language used on this level, SpecTRM-RL, is based on an underlying formal model, so it can be executed and subjected to formal analysis while still being readable to multi-disciplinary team members with minimal training and expertise in discrete math.

The next two levels provide the information necessary to reason about individual component design and implementation issues. Finally, the sixth level provides a view of the operational system. Each level is linked to the levels above and below it. These links provide the relational information that allows reasoning across the hierarchical levels and tracing from high-level requirements down to implementation and vice versa. When working with the system design on a particular level these links provide the rationale and answers to design questions, especially during maintenance activities.

A very important part of an intent specification is that the intent information is present to represent the design rationale upon which the specification is based. This design rationale is integrated directly into the specification so that any new person reading the intent specification for the first time will know why certain design choices were made by another designer. The underlying assumptions supporting the design are also documented at each level. These assumptions are especially important in operational safety analyses. When conditions change such that the assumptions are no longer true, then a new safety analysis is necessary. While a safety analysis document may record these assumptions, the system implementation to which they refer may not have a record of these effects. Without a means for linking these assumptions and how the changes will affect the system overall, a system safety engineer cannot easily determine what the system response will be to a change in this assumption or what part of the system it will most affect [8].

Intent specifications also integrate the safety database and information with the development specification and database so that all necessary information needed by team members in the design phase for design tradeoffs and system analyses is available. When the safety information system is separate from the development specifications and database, designers run the risk of not utilizing safety information at the beginning of the design development process. This leads to information being ignored or argued away as it is too late in the development phase to easily change the design, and any such change would become quite costly. Integrating system safety information into system development at the beginning stages of development ensures that safety is considered throughout the design process and included in analyses and testing from the start [8].

Interface specifications and specification of important aspects of environmental components are also integrated into the intent specification, as are human factors and human interface design. The separation of human-automation interface design from the main system and component design can lead to serious deficiencies in each. Finally, each level of the intent specification includes a specification of the requirements and results of verification and validation activities of the information at that specification level [8].

SpecTRM stands for Specification Toolkit and Requirements Methodology. It is a development environment that allows users to easily create, modify and analyze intent specifications. SpecTRM includes many features important to the intent specification process. First, an empty intent specification in SpecTRM contains headings that, when filled out, help ensure specification completeness. SpecTRM provides this initial intent specification structure to the user to ensure completeness and include system design aspects that may have been ignored otherwise. SpecTRM also provides an easy link creator. Links between levels provide traceability within the specification from the highest requirements all the way down to implementation. This is especially useful for tracking changes in design and rationales and for performing interface testing. Finally, SpecTRM provides various analyses that can be performed on the Level 3 blackbox model [17].

3.1.1 Analyses

SpecTRM currently provides two analyses that can be performed on the individual intent specifications. These are non-determinism and robustness analyses. A model is deterministic if for any given system state and set of inputs, there is only one transition for each state and mode [17]. A model is robust, if for any given system state and set of inputs, a transition exists. That is, a behavior is defined for all possible inputs [17]. These analyses allow the system engineers to eliminate all inconsistencies and incompleteness before the simulation is run. The Level 3 models can be checked automatically for these properties [3].

3.1.2 Simulations

Perhaps the greatest advantage to utilizing intent specifications and SpecTRM is that the models created are executable. The Level 3 blackbox model is a formal representation of an underlying state machine, so the model can be executed given a set of inputs. Individual models can be executed in isolation and multiple models can be executed in an environment in which they interact with each other. Components can be linked to their parent subsystems and the subsystems to the controller to simulate the system-as-a-whole.

Executable models are extremely beneficial to the system development process. These simulations allow system developers to observe the results of interactions between components and the functionality of the subsystem specification and model. This is done at a point in the development lifecycle before code is written or hardware is fabricated, allowing changes to be made and errors to be corrected before costly steps are taken in the system development process. This is also especially important in system software engineering where code maintenance needs to be addressed, particularly in the aerospace industry where maintenance changes to be made can occur after launch. An executable state machine provides the system software maintainers with the ability to incorporate changes to the code from the formal requirements specification, and simulate the effects those changes will have on the rest of the system, again before any code has been implemented or uplinked to the system. This way system engineers can see the affects of system changes on the overall system before any implementation occurs, and potentially hazardous errors can be avoided.

Executable blackbox models help developers to perform trade-off analyses. Engineers can simulate alternative design strategies and determine which approach is most suitable given the constraints and requirements of the system. Instead of a development team needing to create their own platform to perform such trade analyses, the same goal can be achieved by utilizing SpecTRM. Once the trade-off analysis is complete, the development team already has an intent specification for their chosen design in place and ready for more detailed design work.

Lastly, just as different types of system design visualizations can aid in creating an intent specification, different types of visualizations of the underlying state machine can also allow users to facilitate the creation of a mental model of the system's functioning. A high quality mental model of the system will improve the requirements creation and reviewing process [2]. Clearly, having a formal and executable blackbox model of the system provides engineers with the variety of benefits that aid in the proper implementation of a component-based development approach.

3.1.3 SpecTRM-GSC

SpecTRM-GSC stands for SpecTRM-Generic Spacecraft Components. SpecTRM-GSC utilizes the SpecTRM platform to create separate executable intent specifications for different generic spacecraft components. The components are generic to support good practices is spacecraft design reuse. SpecTRM-GSC also contains component-level fault protection, laying the foundation for a fault protection scheme that parallels the spacecraft development process. The formal portion of the SpecTRM-GSC can also be analyzed individually, as part of their parent subsystem, or at the system-level before any implementation has taken place. The creation of such a database of spacecraft components enhances the system design process by providing an existing component framework for beginning spacecraft system design that can be easily tailored to a specific spacecraft's needs through defining the design values that make the component specific to the particular spacecraft being designed.

3.2 Summary

Intent specifications and SpecTRM provide a platform for creating executable models to be utilized in the system development process. Intent specifications capture the design information from system design visualizations and record the rationale behind the design choices made at each step in the development lifecycle. They abstract away the details of design so that the specifications can also be reusable from one project to the next. Various analyses can be performed in SpecTRM, which aids in the development of autonomous systems. System performance can be tested through simulation before any hardware is built or any code is written.

During maintenance, changes to the software can be easily documented and incorporated into the new system. Engineers can also simulate design alternatives for trade-off analyses as well as visualize the underlying state machine to obtain a different perspective of the system. The creation of SpecTRM-GSCs also aids the spacecraft development process by providing complete intent specifications for generic spacecraft components in a reusable and customizable format. The next chapter provides an example of using systems design visualizations to create an intent specification using SpecTRM as applied to a real system.

Chapter 4
MAPS Example

MAPS stands for Mobility and Positioning System. It is a subsystem of the Thermal Tile Processing System (TTPS). The TTPS is a system designed to service the thermal tiles on the space shuttle after the spacecraft has returned to earth. The current process involves humans visually checking each of the tiles and injecting a chemical into them with a handheld tool to waterproof the tiles. This is a tedious process that usually spans the entirety of the three to four months the orbiter is on the ground between missions. In addition, the chemical, DMES, used to waterproof the tiles is extremely toxic to humans, making heavy suits and respirators necessary for completing these tasks while also working in a crowded area [10]. The hope for the TTPS is that it will reduce the need for multiple inspections of individual tiles as the robot will have a much better ability to see tile defects than the naked human eye. It would also eliminate the need for humans to perform this tedious and potentially hazardous task in an area already congested with other workers servicing other parts of the shuttle. The original robot for the TTPS was designed as a research project at CMU with funding from NASA. While it was delivered to NASA in 1994, there is no knowledge of it being used in shuttle servicing operations [10]. The robot design used in this example has been modified from the original for safety and to improve the system example [10].

MAPS was chosen for this case study for two important reasons. First, it is part of a complex system with many links to other subsystems. The system design visualizations described in Chapter 2 can be used to show these links and the system flow into and out of MAPS as it relates to the larger, complex system. Second, isolating one subsystem from the TTPS allows the design details of MAPS to be illustrated in the design visualizations and incorporated into a Level 3 blackbox model. This chapter demonstrates the creation of system design visualizations for MAPS and how the information provided in these visualizations can be used in populating an intent specification in SpecTRM.

4.1 MAPS System Design Visualizations

Chapter 2 outlined seven different system design visualizations. In this section, those visualizations will be used to describe MAPS and to demonstrate their use in the MAPS design. MAPS is one of many subsystems that make up the mobile robot in the TTPS. A preliminary hazard analysis performed at the beginning of the system design phase helps to alert multi-disciplinary team members to potentially hazardous state the system could assume. Identifying these hazards early in the design allows time for engineers to mitigate and even eliminate system hazards. Figure 10 shows a tree diagram for part of a preliminary hazard analysis for MAPS.

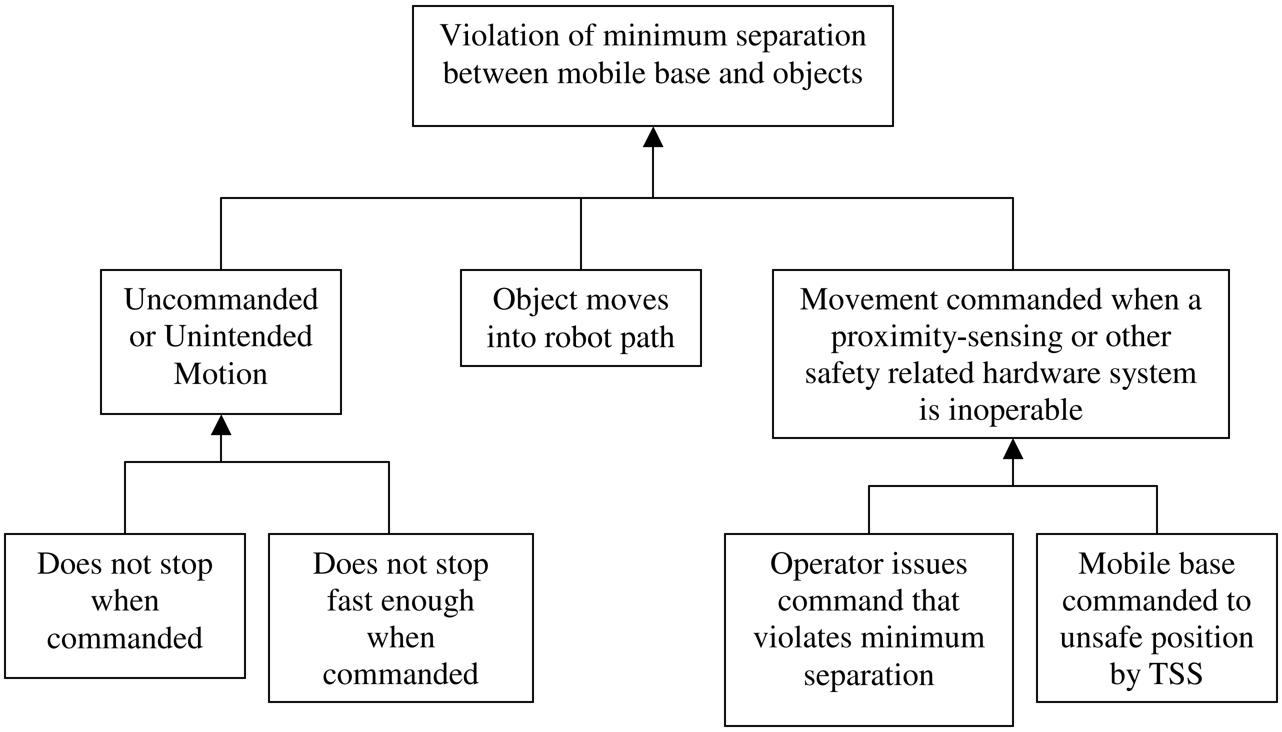

Violation of minimum separation between mobile base and objects
Uncommanded or Unintended Motion
Object moves into robot path
Movement commanded when a proximity-sensing or other safety related hardware system is inoperable
Does not stop when commanded
Does not stop fast enough when commanded
Operator issues command that violates minimum separation
Mobile base commanded to unsafe position by TSS

Figure 10. MAPS Hazard Tree Diagram

This tree diagram shows the factors that could occur and lead to the hazard of violating minimum separation between the mobile base and other objects in the shuttle servicing area. By knowing what factors could lead to the minimum separation violation hazard, engineers can design the system to safely function to avoid this hazard from leading to an accident.

Since MAPS is one of many different subsystems comprising the mobile robot component of the TTPS, it would be good to know what other subsystems are on the mobile robot and how they work together. Figure 11 is a system process flow diagram for the mobile robot to show these interactions in the mobile robot system.

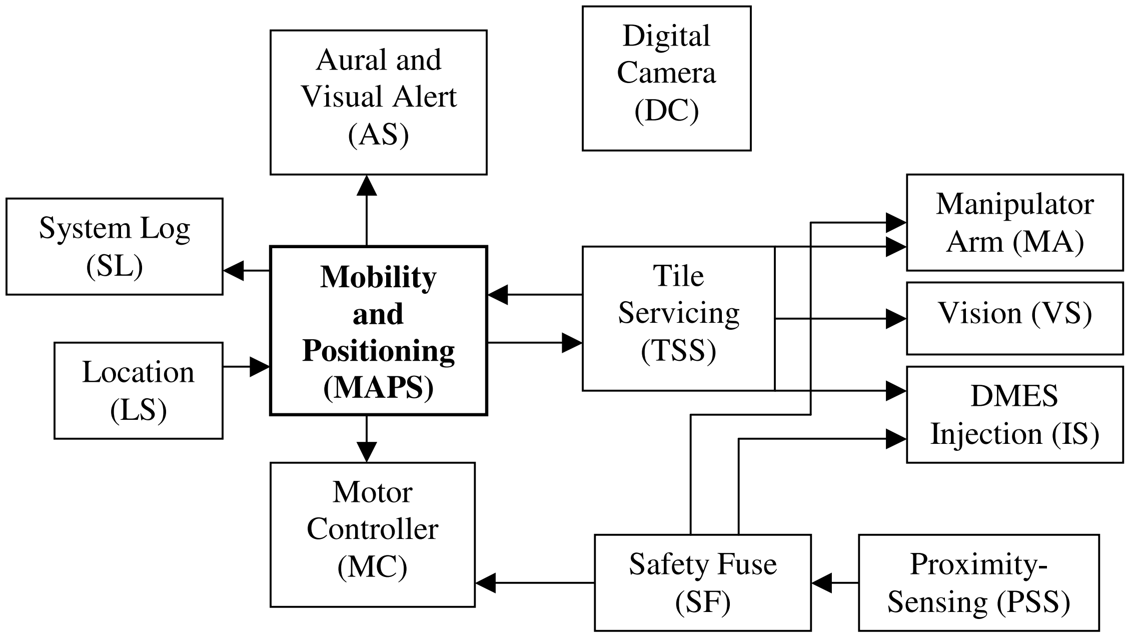


Figure 11. Mobile Robot System Process Flow Diagram

This system process flow diagram shows that MAPS interacts with five other subsystems in the TTPS, the Aural and Visual Alert Subsystem (AS), the System Log (SL), the Location Subsystem (LS), the Motor Controller (MC), and the Tile Servicing Subsystem (TSS). Other subsystems on the mobile robot indirectly affect MAPS through the TSS or through commands issued to other subsystems that are controlled by MAPS. This diagram shows the basic interactions that the MAPS subsystem has during nominal TTPS operations.

An N^2 diagram is another useful visualization to provide more detail about the interactions occurring between the mobile robot subsystems. The N^2 diagram in Figure 12 shows the subsystem interactions directly with MAPS. The subsystems not directly communicating with MAPS were omitted, as the MAPS design is the focus of this system analysis.

Location Subsystem		X			
	TSS	X			
	X	**MAPS**	X	X	X
			Motor Controller		
				Aural and Visual Alert Subsystem	
					System Log

Figure 12. N^2 Diagram for MAPS Subsystem Interactions

This diagram also demonstrates a possible progression for system modeling of the mobile robot. This part of the system is mostly linear in terms of information passed from one subsystem to the next, except for the interactions between MAPS and the TSS. This is a flag to system designers that the integration of these two systems must be carefully designed and tested to prevent interface problems.

The N^2 diagram in Figure 12 can be made larger and more detailed by incorporating more rows and columns between the subsystems and listing each of the different relevant subsystem inputs and outputs. Since the intent specification to be developed from these visualizations is focusing on MAPS alone, it is not necessary to identify all of the inputs and outputs for the subsystems listed in the N^2 diagram. Only the MAPS inputs and outputs are of concern, so they can be better displayed by using an input/output diagram as shown in Figure 13. This diagram shows exactly what information MAPS must provide to other subsystems, or receive from other subsystems in order to function properly within the TTPS. By naming each of the inputs to and outputs from MAPS in Figure 13, more detail is provided about each of these variables.

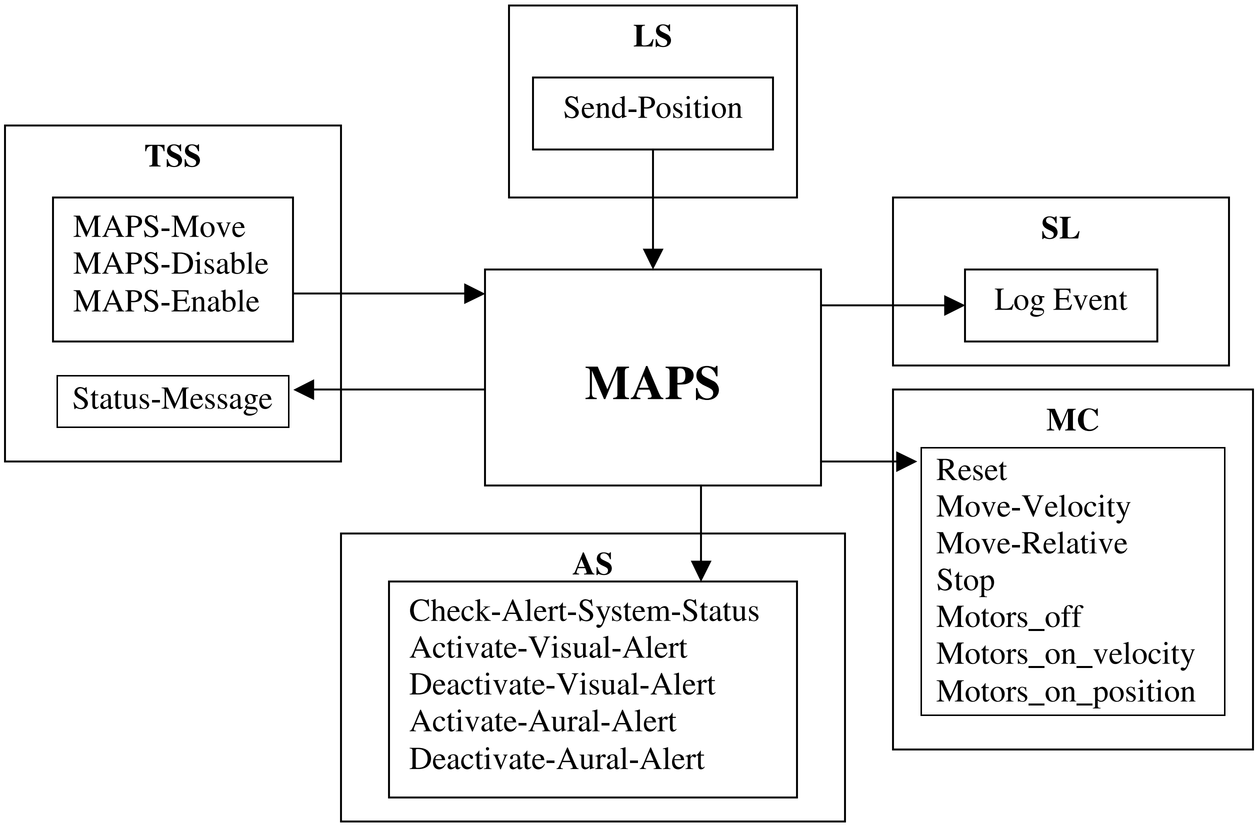


Figure 13. MAPS Input/Output Diagram

MAPS is responsible for issuing the movement commands to the MC as supplied to MAPS either by the operator or by the TSS. To accomplish its tasks, MAPS must have operating modes that allow for safe operation depending on what MAPS is currently trying to accomplish and what the state of the entire TTPS is at the time. The MAPS movement control operating modes are Initialization, Computer Mode, Operator Mode, and Halted Mode. Each of these modes has different system constraints when in that operating mode and for transferring from that operating mode to another. One manner of displaying information about these modes is in a key metric chart. The chart describes the requirements for transferring to each of the movement control modes, as well as from which modes transfer is allowable. Safety constraints and rationales for each of the modes and mode requirements can also be included, as is seen in Figure 14. By displaying the operating modes and the requirements in this way, engineers can see how the system can change modes and what problems can occur if not all transitions are properly designed in the system.

Mode	Mode Requirements	Safety Constraint
Initialization Mode	Enter Initialization at Powerup Enter Initialization after Safety Fuse reset	System must reinitialize after a safety fuse reset event to prevent previously uncompleted movement from being continued, putting the mobile robot into a possible hazard situation
Operator Mode	Enter Operator Mode from Initialization Enter Operator after a temporary shutdown Enter Operator after deadman switch is released Enter Operator after Human Operator command to switch to Operator	The joystick must be connected and in the neutral position to enter Operator mode The deadman switch must be depressed to enter Operator mode MAPS will default to Operator mode after Initialization to ensure the robot is not moved using automation without notice to the rest of the staff in the shuttle maintenance area While in Operator mode only movements from the joystick will be recognized, all TSS commands will be ignored
Computer Mode	Enter Computer after Human Operator commands to switch to Computer	All joystick commands while the robot is in Computer mode will be ignored so as not to interfere with TSS and MAPS commands or cause inadvertent robot movement Release of the deadman switch will allow the human operator to immediately halt robot movement and switch to Operator mode
Halted Mode	Enter Halted when Safety Fuse enters Halt State	Other subsystems may be still operating while the movement control mode is in Halted, so the movement control system must only switch to Initialize after Halted to allow the system and the human operator to know the current system state

Figure 14. MAPS Movement Control Mode Key Metric Chart

Now that the movement control modes are known, and rules for entering each mode are documented, a state transition diagram can be created to better visualize the state changes and be sure that only one transition from one mode to another is possible. The state transition diagram in Figure 15 shows the designed mode transitions for MAPS.

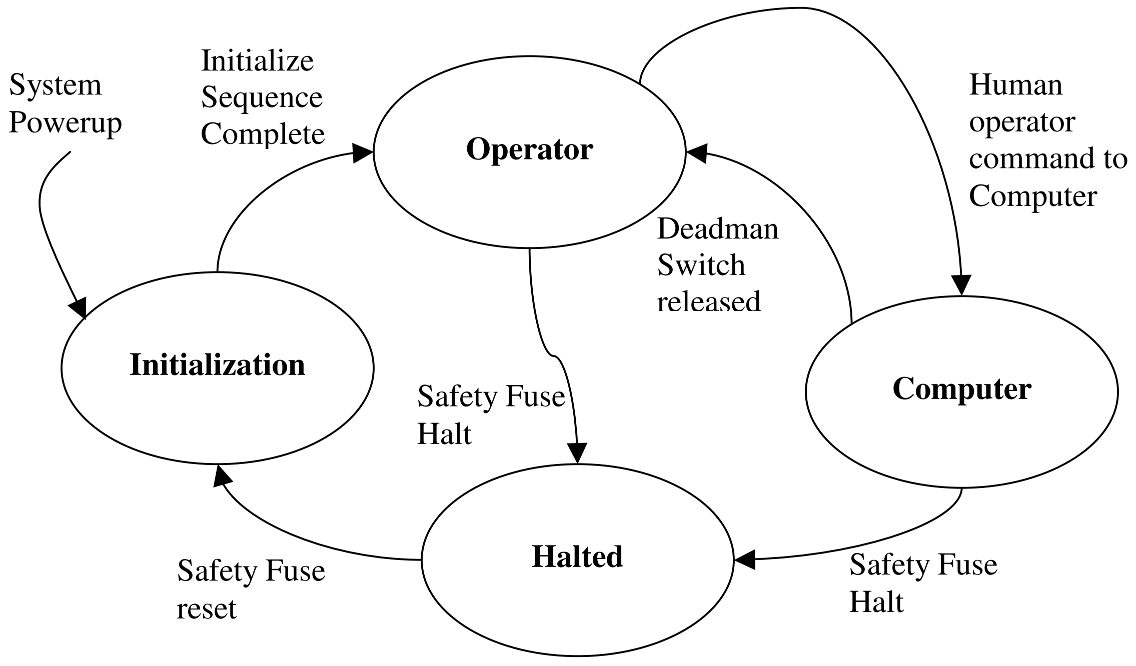


Figure 15. MAPS Movement Control State Transition Diagram

Now it is easier to see that once the system enters Halted mode, it is only possible to get out of that mode by going to Initialization. This is important to prevent mobile robot movement while there may be a problem in one of the subsystems, or while there are problems in the operating environment of the robot.

Mode confusion is a relatively new problem in the aerospace industry as more and more systems are adding automation and making the human operators into system monitors, rather than the humans operating the system directly. When a system relies on humans to monitor, but not interactively operate a system, a new type of error can occur, known as mode confusion. Mode confusion is when an automated system enters a mode without the human operator knowing that a change has occurred. The human operator may know that the mode has changed, but does not know what mode the system is now in, or he may not know what caused the mode change to occur, or in the worst case scenario, the operator may not know that a mode change has occurred at all, and still believes the system to be operating in a different mode. This kind of confusion can lead to serious accidents. User and task models help engineers to recognize if a potential for mode confusion exists in a system so that design changes may be made to prevent it. Figure 16 shows a user and task model for MAPS describing the situations that cause MAPS to stop the mobile robot.

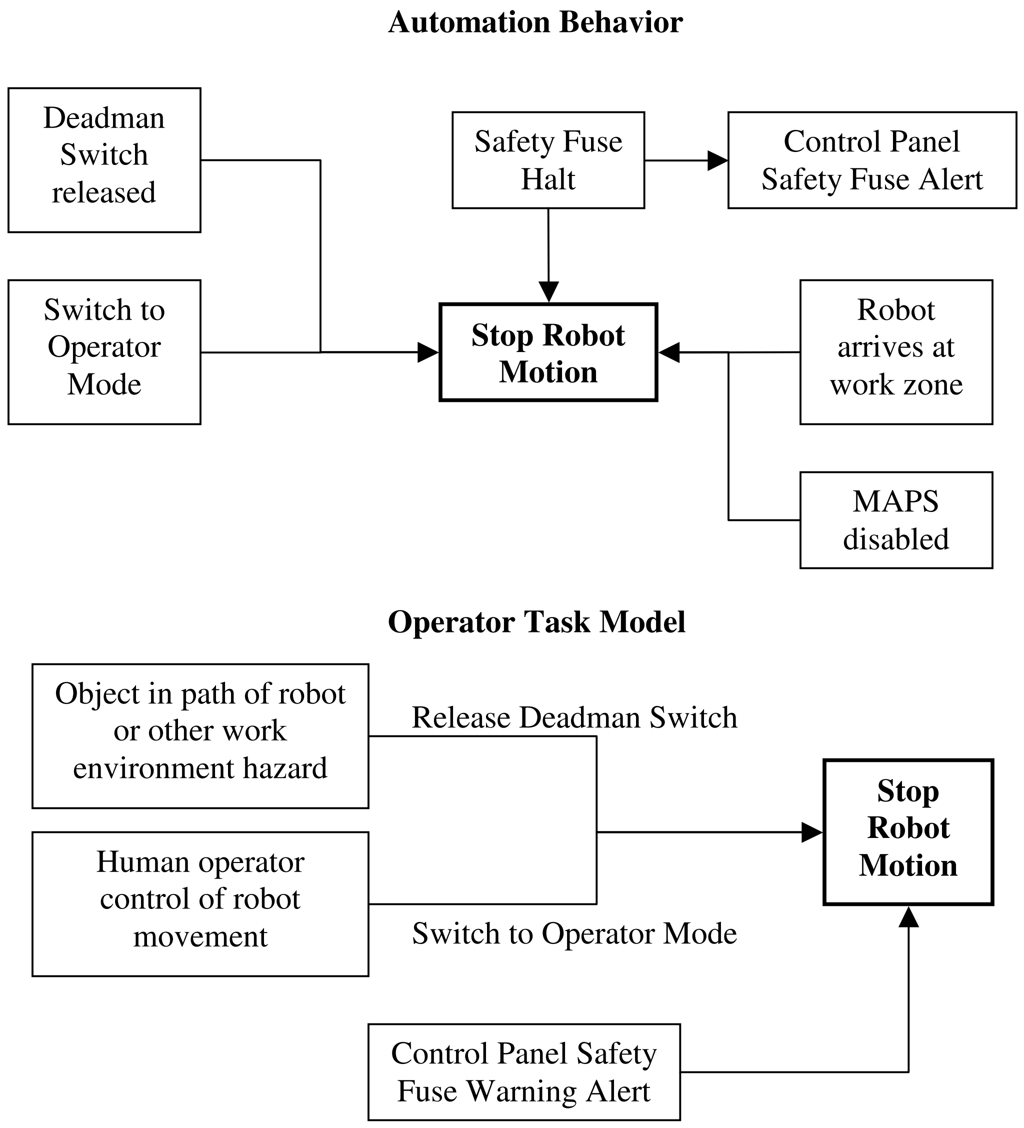

Automation Behavior
Deadman Switch released
Switch to Operator Mode
Safety Fuse Halt
Control Panel Safety Fuse Alert
Stop Robot Motion
Robot arrives at work zone
MAPS disabled
Operator Task Model
Object in path of robot or other work environment hazard
Release Deadman Switch
Human operator control of robot movement
Switch to Operator Mode
Stop Robot Motion
Control Panel Safety Fuse Warning Alert

Figure 16. MAPS User and Task Model

The task model shows the five different situations that could occur to cause MAPS to stop the mobile robot. In the user model, the operator is directly responsible for two of the actions: releasing the deadman switch on the control panel and switching MAPS to Operator mode. A Safety Fuse to the Halt state alert would also be displayed to the operator on the control panel, so the operator would know if the robot was stopped for this reason. The operator will not know, however, if the mobile robot stops because it has reached the work area it was directed to or if it stops because MAPS was disabled. This problem can lead to significant work delays if the operator is unsure of what to do to get the robot to begin moving again, or if any action on the human operator's part is required at all. It could also lead to a hazard if the operator tries to command the robot to move but it has begun servicing a tile in its commanded position. The user and task models bring this discrepancy to the attention of the design team so that the problem can be dealt with in the design. Instead of having a problem such as this, a simple solution such as displaying a status message to the operator that MAPS has been disabled or that the robot has arrived at its location can be incorporated into the design instead.

These visualizations were used to create an intent specification for MAPS in the SpecTRM environment. The next section describes how the data from these visualizations can be easily translated into SpecTRM-RL and used to populate Levels 1, 2 and 3 of an intent specification.

4.2 SpecTRM-RL Examples of Visualization Information

This section describes how the information on MAPS presented in the system design visualizations can be used to synthesize an intent specification in SpecTRM to create an executable system model based on requirements and design specifications. Appendix A provides complete Levels 1, 2 and 3 of the MAPS intent specification.

Level 0 of the specification was left blank in the MAPS specification, as it is particular to the organization and engineering team of the project. Since a project management group does not exist for this project, Level 0 would be filled in once it is determined. Level 1 of the specification includes the system-level goals, requirements and constraints. The system process flow in Figure 11 identifies the subsystems that need to be included in the intent specification and have high-level functional requirements, even if the detailed design is not focusing on that subsystem's function. For example, Figure 17 demonstrates a high-level functional requirement for the TSS, based on the subsystem relationships shown in the system process flow diagram in Figure 11, as well as the N^2 diagram in Figure 12.

[TSS-FR2.2] When the mobility and positioning system is being commanded by the on-board TSS, it shall provide the route of travel and a destination. Locations shall be provided in world coordinates. [→1.4.2.2] [↓2.1.1] [↓2.4.2.6] [↓2.4.5.3]
Rationale: For safety reasons, the operator is responsible for mobile base movement. However, to allow for potential autonomy and thus more efficient operation, the TSS can directly issue movement commands to the mobility and positioning subsystem. Any TSS movement commands must be monitored by the operator.

Figure 17. Level 1 Functional Requirement

This specifies part of the relationship of the TSS to MAPS at a high-level. This requirement also includes four links at the end. These are hyperlinks within SpecTRM to link different parts of the intent specification for design traceability. The first link points to the subsystem goal, also at Level 1. Goals are included for documentation and intent completeness, but are too high level to be requirements. They are not necessarily testable, and there may be many system designs that meet the system's goals that are unacceptable due to other constraints. High-level requirements are generated from the goals and constraints present at this level. These are the "shall" statements that specify what the system is to do and are the testable requirements.

The remaining three links point down to design principles at Level 2 that provide information about the route determination and coordinates for robot movement. The rationale also informs the engineering design team as to why this relationship exists and what its intended role is in the entire system. In this example, the rationale specifies the need to alert the human operator of the movement commands issued by the TSS to avoid the potential mode confusion problem identified by the MAPS user and task model in Figure 16. The information from the user and task model is captured in this rationale on Level 1 and will also be considered when applying this high-level functional requirement to a design principle on Level 2.

These high-level functional requirements and constraints can also be found in key metric charts, such as the one for the MAPS movement control modes. An example safety-related design constraint can be taken from the information in Figure 14 and can be seen in Figure 18.

[MAPS-SC1] The mobile base must move only when commanded by the operator or when commanded by the TSS and approved by the operator. [→H1] [→MAPS-FR2.5]
Rationale: The mobile robot must not move if there are objects in its path or if its subsystems are not stowed for movement.

Figure 18. Level 1 Safety-Related Design Constraint

The links at the end of the safety constraint point to the hazard this constraint is to mitigate and the functional requirement it constrains. This safety constraint is expanded in sub-constraints to include in detail the different ways that MAPS must stop the robot from moving to prevent a hazard. These sub-constraints of the MAPS-SC1 safety constraint can be found in Appendix A.

Another part of Level 1 in SpecTRM is information on the preliminary subsystem hazard analysis. A hazard analysis at this level involves identifying system hazards, defining a safety policy, classifying the hazards, and performing a hazard analysis. The hazards in the TTPS relate to human injury or damage to the orbiter. The identified hazards must be eliminated or mitigated by the system design. If a hazard cannot be eliminated, then the design features and development procedures that were used to reduce the hazard level must be documented and presented to the system management and customers before the system development process can resume. Testing is not enough to mitigate a hazard because it is impossible to know if all possible occurrences or situations to cause a hazard are known, before the hazard actually occurs.

A fault tree is one common way of performing a hazard analysis, and a portion of the MAPS fault tree was shown in Figure 10. The information in this tree diagram can be used to create a hazard log entry in SpecTRM. Figure 19 shows the hazard log entry created using the tree diagram. Level 1 information shown in the hazard log description consists of general background, historical information, environment descriptions, assumptions and constraints, system functional goals, operator requirements, interface requirements, design and safety constraints and information about the verification and validation requirements and results on the information at this level. There are also hyperlinks to each of these references.

H1: *Violation of minimum separation between mobile base and objects (including orbiter and humans.)*

Subsystem: MAPS, vision system, proximity sensing system, motor controller location system, visual and aural alert system, operator displays and controls
Operation/Phase: movement from one work zone to another
High-Level Casual Factors:

Uncommanded or unintended motion;
Not stopping when commanded or not stopping fast enough;
Operator issues command that violates minimum separation between robot and object;
Mobile base commanded to unsafe position by TSS;
Movement commanded when a proximity-sensing or other safety related hardware system is inoperable;
Object moves into robot path.

Level and Effect: A1-2
Safety Constraints:

Mobile base must move only when commanded [←MAPS-SC1] [←MAPS-FR2.4];
Mobile base must stop when commanded [←MAPS-SC2] [←SF-FR2] [←SF-FR1];
Mobile base must not be commanded to an occupied position [←MAPS-SC2];
The operator shall supervise all robot base movement [←OP1];
The operator shall authorize all robot base movements before they are performed, including movements commanded by the TSS [←OP2] [←Con5];
The operator must have information about objects in robot path and the ability to stop the mobile base within 0.5 sec [←OP2] [←OP3] [←Disp1] [←Con5] [←Con6];
Movement warnings must be provided [←MAPS-FR6] [←MAPS-SC9] [←AS-FR1];
Mobile base must not move if any safety-related subsystem is not operational [←MAPS-SC8] [←SF-FR3] [←SF-FR4];
The motor controller must be able to stop the motion of the mobile base within 0.2 seconds of receiving a STOP command [←MC-SC1];
The maximum velocity of the robot must be no more than 30cm/sec [←MC-SC2];
The proximity sensing subsystem shall send a signal to the safety fuse in the event of contact with any external object [←PSS-FR1];
It must be possible to move the mobile base out of the way manually in case of an emergency [←MB-SC3];

Analyses Performed:
Actions Taken:
Status:
Verification:
Final Disposal (Closeout Status):
Responsible Engineer:
Remarks:

Figure 19. Level 1 Hazard Log Description

Level 2 of the intent specification specifies the design principles used to implement the

Level 1 requirements. Figure 20 provides an example of a design principle taken from the MAPS movement control key metric chart in Figure 14 and state transition diagram in Figure 15 and linked to a safety-related constraint from Level 1 (Figure 18). It defines the conditions for entering one of the movement control modes in MAPS.

> [2.4.7.1] **Initialization Mode:** MAPS enters initialization mode on powerup and after the safety fuse is reset. [↑MAPS-SC1.5]
> *Rationale: When the safety fuse signals a problem with the robot, there is likely to be some type of repair action changing the state of the robot and therefore requiring the entire initialization sequence and ensuring the robot is in a safe state before movement.*

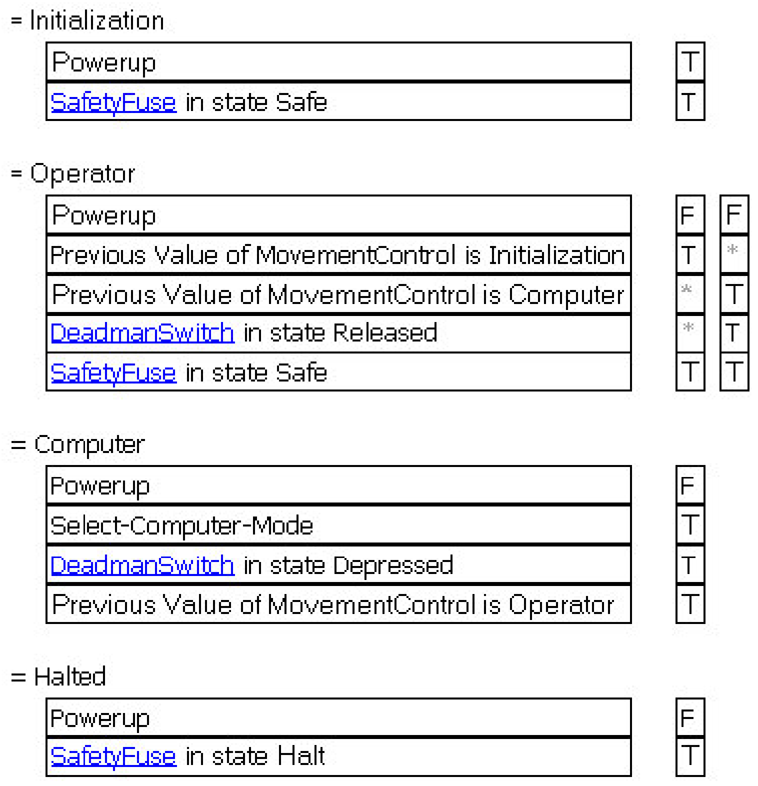
Figure 20. Level 2 Design Principle

Level 3 of the intent specifications contains a formal, blackbox model of the component's externally visible behavior. The formal models, which are based on state machines, are specified using a language called SpecTRM-RL that was designed with reviewability and ease of learning as goals. Experience in using the language on industrial projects shows engineers can learn to read SpecTRM-RL models with about ten to fifteen minutes of training. This allows for formal models based on the system requirements to be easily constructed and analyzed prior to hardware and software development. The resulting document is in English and is easily readable by multi-disciplinary design team members.

The behavior of MAPS, or the logic for determining subsystem function, sending outputs to other subsystems, and changing the inferred operating state, is specified using a tabular notation called AND/OR tables. The rows of the tables indicate AND relationships, while the columns represent ORs. This logic can be used to represent state and mode transitions, such as those shown in the MAPS movement control state transition diagram in Figure 15. Figure 21 shows the transition conditions required for the "MovementControl" supervisor mode to take the values "Initialization," "Operator," "Computer" and "Halted" as they were described in Figure 15. Using the AND/OR tables in Figure 21, the MAPS operating mode "MovementControl" will transition to a different mode if any of the columns in the transition table evaluate to true. In other words, if the "MovementControl" mode was previously Initialize and the "Safety Fuse" is in the state "Safe" and "Powerup" is False, then "MovementControl" will transition to

"Operator." "MovementControl" will also transfer to "Operator" if the "Safety Fuse" is still in the state "Safe" but now "Deadman switch" is in the state "Released" and the previous mode of "MovementControl" was "Computer."

As seen in the four transition tables, there are several statements with an asterisk in the OR column. This represents a "don't care" condition. In the example in Figure 21, the control mode "doesn't care" what the switch position is if it is not coming from the "Computer" mode. A separate alert and control will ensure that the switch is pressed in case it needs to be released again to change the "MovementControl" mode again.



= Initialization

Powerup	T
SafetyFuse in state Safe	T

= Operator

Powerup	F	F
Previous Value of MovementControl is Initialization	T	*
Previous Value of MovementControl is Computer	*	T
DeadmanSwitch in state Released	*	T
SafetyFuse in state Safe	T	T

= Computer

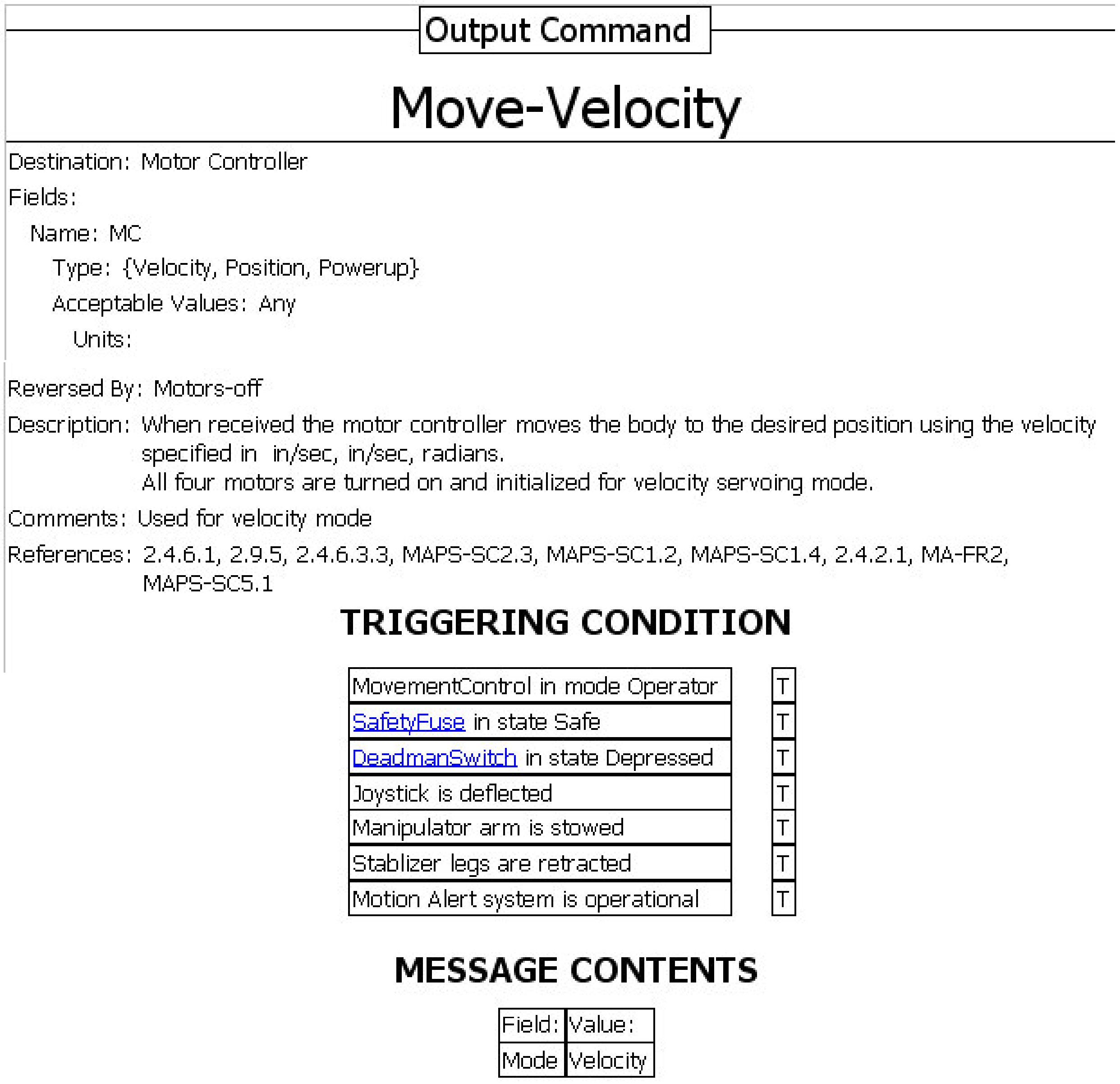
Powerup	F
Select-Computer-Mode	T
DeadmanSwitch in state Depressed	T
Previous Value of MovementControl is Operator	T

= Halted

Powerup	F
SafetyFuse in state Halt	T

Figure 21. Level 3 And/Or Table

Since the blackbox model is based only on the system's externally visible behavior, inputs and outputs are other main SpecTRM-RL elements. The input/output diagram for MAPS shown in Figure 13 describes the major inputs to and outputs from MAPS. Information from this diagram can be used to create the Level 3 inputs and outputs for the intent specification. Figure 22 shows an example output from the MAPS input/output diagram. The Move-Velocity output from MAPS to the motor controller is the signal that causes the MC to begin robot motion. When the triggering condition shown by the AND/OR table in Figure 22 is met, the Move-Velocity command is issued and the MC begins robot motion.



Output Command

Move-Velocity

Destination: Motor Controller

Fields:

Name: MC

Type: {Velocity, Position, Powerup}

Acceptable Values: Any

Units:

Reversed By: Motors-off

Description: When received the motor controller moves the body to the desired position using the velocity specified in in/sec, in/sec, radians.
All four motors are turned on and initialized for velocity servoing mode.

Comments: Used for velocity mode

References: 2.4.6.1, 2.9.5, 2.4.6.3.3, MAPS-SC2.3, MAPS-SC1.2, MAPS-SC1.4, 2.4.2.1, MA-FR2, MAPS-SC5.1

TRIGGERING CONDITION

MovementControl in mode Operator	T
SafetyFuse in state Safe	T
DeadmanSwitch in state Depressed	T
Joystick is deflected	T
Manipulator arm is stowed	T
Stablizer legs are retracted	T
Motion Alert system is operational	T

MESSAGE CONTENTS

Field:	Value:
Mode	Velocity

Figure 22. Level 3 Output Command

SpecTRM has other elements that allow the blackbox behavior to be modelled. Functions in SpecTRM-RL are written in an Ada-like programming script. This script allows users to calculate values from system inputs. It can be used to calculate the duration that a variable is valid or to calculate an exact value of a subsystem variable. Another SpecTRM element known as a macro allows users to abstract common logic and to increase readability. A macro can take a piece of an AND/OR table from another part of the model and give it a name. This name can then be substituted for that table portion elsewhere in the model. The macro definition is a single AND/OR table. This table will evaluate to true or to false. When the table is true, the macro as a whole evaluates to true where it is used in other model elements. Similarly, if the table is false, then the macro evaluates to false.

The elements of the SpecTRM blackbox include output commands, output values, modes, states, macros, functions, command inputs and input values. Other information contained at Level 3 includes communication channels, operational procedures, user models and the results of performing various analyses and simulations on the blackbox model. The blackbox model for MAPS can be found in Appendix A.

4.3 Summary

MAPS is a case study for using system design visualizations to synthesize an intent specification. The subsystem interactions with other parts of the TTPS make MAPS a good representative system to illustrate how visualizations describing these interactions and design details can be used to create an executable and analyzable model in SpecTRM. An intent specification was created for the MAPS subsystem. Rationale was captured at each level of development and links provide traceability throughout the entire document. The blackbox system behavior was specified to allow further system level analysis and future simulation of MAPS together with the other subsystems that comprise the mobile robot part of the TTPS.

Chapter 5
Conclusion

An increase in system complexity is a major obstacle for the aerospace industry, and any other technology driven industries. This complexity is a result of the combination of implementing emerging technologies, particularly automation, and advanced industry goals. Software has become an integral part of complex system design, especially in spacecraft. Software now controls not only the spacecraft hardware, but also the onboard science and operations. It was hoped that trading in the simplicity of the previously used hardware designs for the flexibility and abilities of software would decrease system complexity, but the reverse has been seen to happen. Software can provide a solution for some design problems, however the lack of software integration throughout the system development process and the lack of understanding of the full workings of the system software have made today's spacecraft even more complex.

Returning to the hardware dependent predecessors is not a solution to this problem. As space exploration goals grow more ambitious, so does the need for autonomous spacecraft control systems. Artificial intelligence has not yet reached a development level where scientists are willing to trust spacecraft control completely in an automated system. This means incorporating a human operator in the system, and increasing the complexity yet again by having to determine and define the human-machine relationship in the system.

In addition to the increasing system complexity, the aerospace industry faces many other engineering challenges. The diverse education and experience of multi-disciplinary system design teams creates a large communication problem. Engineers working on different subsystems use different tools and languages to express their system needs and capabilities, with little overlap. Miscommunication of key information for subsystem interfaces during the system design phase can lead to costly errors later in system development. The poor usage of requirements and specification documentation in the system design phases also leads to system design problems. These documents are typically not made available to the design engineers performing the detailed subsystem design, nor are they in an easily usable and readable format. If the documents are used, the rationale behind the decisions is often missing, making design changes difficult and time consuming. This is even more difficult if the problem is not detected early in the development lifecycle. There needs to be a common medium through which engineers from different disciplines can communicate clearly and effectively and successfully analyze and execute the system level design prior to hardware manufacturing and software coding.

The current economy does not allow for spacecraft system engineering to exist as it once did. Budget cuts and public disinterest in space exploration have made mission funding and staffing much more difficult. Where shortcuts were taken, the price was paid, and several successive spacecraft failures were greatly publicized. While the budgets for these spacecraft were relatively low, the failed missions were still lost dollars, and resulted in a lack of public support for the continuation of space exploration. The aerospace industry must now find a way to change its approach to spacecraft engineering in order to produce successful spacecraft and exploration missions under tight budget constraints.

This thesis proposed a different approach to spacecraft system development to address the needs of the aerospace industry. The approach uses system design visualizations already used by multi-disciplinary design teams to synthesize the requirements and specifications for the system. These requirements and specifications form an intent specification that includes design decisions and the rationales behind them. In this approach, subsystem designers take part in the requirements and specification generation, and are not just bound by them.

The intent specification is created in a systems engineering development environment known as SpecTRM. SpecTRM is a toolkit that allows users to create intent specifications that assist engineers in managing the system requirements, design and development process. An intent specification provides more structure than a plain requirements document and makes the information contained more readable and accessible to project participants. By utilizing intent specifications, engineers can detect specification errors early in system development so that changes can be made and errors fixed before leaving the design phase. Engineers can also trace the design choices made in the development lifecycle and detail the rationale behind the choices made.

This thesis provided an example of utilizing system design visualizations to create an intent specification as applied to the MAPS subsystem of the Thermal Tile Processing System. MAPS was chosen as the test case for evaluating this method for synthesizing intent specifications because it is part of a complex system with a high level of subsystem interactions, and it can be modeled separately from the rest of the system for subsystem level details. System design visualizations for different aspects of MAPS were created to demonstrate the use of visualizations in system design. The information from those visualizations were then used to create a MAPS intent specification for formal analysis.

Through applying this methodology on MAPS, it was shown how many of the problems outlined in Chapter 1 can be solved. The use of SpecTRM helps to solve the problems of design decision traceability through the recording of rationale at every stage of development. In addition, the use of SpecTRM-RL at Level 3 of the intent specification provides a readable and unambiguous formal specification that provides a common language with which engineers can easily communicate their requirements specifications. This blackbox model of MAPS in Level 3 of SpecTRM is based on a state machine model that does not actually require a background in discrete mathematics to create. A minimal amount of training is necessary to create SpecTRM models that can be formally analyzed and executed.

An important benefit of using this approach to system design is the rigorous development of requirements specifications. The intent specification created by this process can be thoroughly analyzed for major inconsistencies and incompleteness prior to the next development stage. In addition, system functions can be easily simulated and the results of these simulations analyzed for deficiencies in the requirements specifications. This enables many errors and inadvertent omissions to be found early in the system development lifecycle. Errors and other problems found in this phase are far easier and less costly to fix than those found once implementation has begun. In addition, the overall system is designed from a safety standpoint with a goal of eliminating hazards wherever possible, rather than having to justify the risk of a hazard occurring if an error is found later in the system development lifecycle.

The material presented here on utilizing system design visualizations to synthesize intent specifications demonstrates the potential for application in complex system design. Any industry driven by developing technology, especially with an emphasis in adding automation to system implementation, can benefit from using this approach to link a formal approach to requirements specifications with existing system design techniques. The results of this thesis merit further investigation into the use of system design visualizations in synthesizing intent specifications.

References

[1] Casani, John. *Report on the Loss of the Mars Polar Lander and Deep Space 2 Missions*. Jet Propulsion Laboratory. Pasadena, CA. 22 March 2000.

[2] Dulac, Nicolas, Thomas Viguier, Nancy Leveson, and Margaret-Anne Storey. "On the Use of Visualization in Formal Requirements Specification." *Proceedings of the International Conference on Requirements Engineering*. Essen, Germany. September 2002.

[3] Heimdahl, Mats P.E. and Nancy Leveson. "Completeness and Consistency Analysis of State-Based Requirements." *IEEE Transactions on Software Engineering*. May 1996.

[4] International Council on Systems Engineering. "Systems Engineering." Online. 4 June 1999.

[5] Larson, Wiley J. and James R. Wertz editors. *Space Mission Analysis and Design.* Second Ed. California. Microcosm, Inc. 1992

[6] Leveson, Nancy, L. Denise Pinnel, Sean David Sandys, Suichi Koga, Jon Damon Reese. "Analyzing Software Specifications for Mode Confusion Potential." *Presented at the Workshop on Human Error and System Development*. Glasgow. March 1997.

[7] Leveson, Nancy. "Completeness in Formal Specification Language Design for Process-Control Systems." *Proceedings of Formal Methods in Software Practice Conference*. Portland, OR. August 2000.

[8] Leveson, Nancy. "Intent Specifications: An Approach to Building Human-Centered Specifications." *IEEE Transactions on Software Engineering*. January 2000.

[9] Leveson, Nancy. *Safeware : System Safety and Computers.* Addison-Wesley Publiching Company Inc. 1995

[10] Leveson, Nancy. "Shuttle Thermal Tile Processing Example Intent Specification." 27 July 2002.

[11] Leveson, Nancy. "Systemic Factors in Software-Related Spacecraft Accidents." *Space 2001 Conference.* August 2001.

[12] Lions, J.L., et al. *Ariane 5 Flight 501 Failure: Report by the Inquiry Board*. Paris, France. 19 July 1996.

[13] Massachusetts Institute of Technology. The ESD Symposium Committee. *ESD Terms and Definitions*. Version 12. 19 October 2001.

[14] Massachusetts Institute of Technology. Space Systems Engineering, course 16.89. "Course Notes and Design Document" Spring 2003.

[15] National Aeronautics and Space Administration. *NASA Systems Engineering Handbook.* June 1995.

[16] Rodriguez, M., et al. "Identifying Mode Confusion Potential in Software Design." *Digital Aviation Systems Conference.* October 2000.

[17] Safeware Engineering Corporation. *SpecTRM User Manual*. Version 1.0.0. 2003.

[18] Stephenson, Arthur G., et al. *Mars Climate Orbiter Mishap Investigation Board Phase I Report*. 10 November 1999.

[19] Ulrich, Karl T. and Steven D. Eppinger. *Product Design and Development.* McGraw-Hill, Inc. 1995.

[20] Van Vliet, Hans. *Software Engineering Principles and Practice*. Second Ed. New York: Wiley, 2001.

[21] Website. http://education.aero.org/images_ver3/n2chart.pdf

[22] Weiss, Kathryn A. "Component-Based Systems Engineering for Autonomous Spacecraft" Master's Thesis. Massachusetts Institute of Technology, Cambridge, MA. August 2003.

[23] Zimmerman, M.K., Kristina Lundqvist, Nancy Leveson. "Investigating the Readability of Formal Requirements Specifications Languages." *International Conference on Software Engineering.* May 2002.

Appendix A
MAPS Intent Specification

Level 0: Program Management Information

This section contains pointers to the program management documents for this project.

Program Management Plans

This section contains pointers to the software and hardware development plans and other management information.

The following is an outline from IEEE Standard 1058 (dated 1987) for software project management plans:

1. Introduction
    - 1.1 Project Overview
    - 1.2 Project Deliverables
    - 1.3 Evolution of the Software Project Management Plan
    - 1.4 Reference Materials
    - 1.5 Definition and Acronyms

2. Project Organization
    - 2.1 Process Model
    - 2.2 Organizational Structure
    - 2.3 Organizational Boundaries and Interfaces
    - 2.4 Project Responsibilities

3. Managerial Process
    - 3.1 Management Objectives and Priorities
    - 3.2 Assumptions, Dependencies, and Constraints
    - 3.3 Risk Management
    - 3.4 Monitoring and Controlling Mechanisms
    - 3.5 Staffing Plan

4. Technical Process
    - 4.1 Methods, Tools and Techniques
    - 4.2 Software Documentation
    - 4.3 Project Support Functions

5. Work Packages, Schedule and Budget
    - 5.1 Work Packages
    - 5.2 Dependencies
    - 5.3 Resources Requirements
    - 5.4 Budget and Resource Allocation
    - 5.5 Schedule

System Safety Plan

The updated and current System Safety Plan would be included here or a pointer to it would exist here. Within this plan there will be pointers to the various parts of the intent specification that implement the parts of the plan, such as the hazard list, various hazard analysis results etc.

The system safety program plan (SSPP) is a management document that describes the system safety objectives and how they will be achieved. It provides a regulatory agency, contracting agency, or manager with a baseline (a basis of understanding among multiple groups and within a single group at a time) with which to evaluate compliance and progress. Specifying a plan should be the first step in any safety-critical project.

Plans for subsystem safety should be part of the SSPP rather than in separate documents. Many standards exist for program plans - each is a little different, but all contain similar information. Devising a general plan for everyone is not practical, however: Each plan needs to be tailored to its project and golas and to fit the corporate culture and management system.

Although one plan will not suffice for all types of projects, the following information might be included in a comprehensive plan:

I. General Considerations
    A. Introduction
        B. Scope and Purpose
    C. Objectives
    D. Applicable Standards
    E. Progress Reporting
    F. Documentation and Reports

II. System Safety Organization
    A. Personnel Qualification and Duties
    B. Functional Organization
    C. Staffing and Manpower
    D. Communication Channels
    E. Responsibilty, Authority and Accountability
    F. Subcontractor Responsibilities
    G. Coordination
    H. System Safety Groups/ System Working Groups
    I. Safety Program Interfaces with Other Disciplines
        Reliability
        Maintainability
        Design and Software Engineering
        Software Development
        Configuration Management
        Quality Assurance
        Human Factors
        Test
        Industrial Safety

III. System Safety Program Schedule
    A. Critical Checkpoints and Milestones
    B. Start and Completion Dates of Tasks, Reports, Reviews
    C. Review Procedures and Participants

IV. System Safety Criteria
    A. Definitions
    B. Start and Completion Dates of Tasks, Reports, Reviews
    C. Review Procedures and Participants
        Hazard Severuty Categories
        Hazard Probability Levels
        Risk Assessment
    D. System Safety Procedure
    E. Safety Design Criteria
        Hardware

Software

F. Special Contractual Requirements

V. Safety Data
    A. Data Requirements
        Deliverable
        Non-Deliverable
    B. Hazard Tracking and Reporting System
    C. Requirments and Use of Safety Data
        Hazard Data Collection
        Lessons Learned
        Documentation and Files (Safety Data Library)
        Records Retention
VI. Hazard Analyses (Types, Documentation and Expected Uses)
    A. Preliminary Hazard Analysis
    B. System Hazard Analyses
    C. Subsystem Hazard Analyses (including Software Hazard Analyses)
    D. Operating System Hazard Analyses
    E. Integration of Subcontractor Analyses with Overall System Hazard Analyses
    F. Tracing of System Hazards into Subsystems
VII. Verification
    A. Safety Related Testing
    B. Special Demonstrations
    C. Review and Feedback Procedures
VIII. Audit Program
IX. Operations
    A. Emergency and Contingency Procedures
    B. Configuration and Control Activities
    C. Training
X. Hazard and Incident Reporting and Investigation During Operations
XI. Special Safety Activities
    A. Range Safety
    B. Facility Safety
    C. Explosives Safety
    D. Nuclear Safety
    E. Chemical and Biological Safety

Safety Policy

The Safety Policy of the Thermal Tile Processing System, and the MAPS subsytems, will be in accordance with the Safey Policies of the agencies that will use this system, namely NASA Dryden Flight Research Center or NASA Kennedy Space Center.

Level 1: System-Level Goals, Requirements, and Constraints

In this example, some arbitrary choices are made about what information to put at Level 1-this specification is only an example ad there is no real customer and thus no way to determine what is needed in the customer model. Some of the information included here may be more appropriate at Level 2 for some systems and vice versa.

Introduction

(The following description is adapted from *A Mobile Robot System for Ground Servicing Operations on the Space Shuttle* by K. Dowling, R. Bennett, M. Blackwell, T. Graham, S. Gatrall, R. O'Toole, and H. Schempf. The original Tessellator robot was designed as a research project in the Robotics Dept. at CMU with NASA funding. This specification was derived from one that students pursuing a master's degree created for a project at the SEI. Changes have been made from the original specification in order to satisfy our different goals.)

The Thermal Tile Processing System (TTPS) is designed to service tiles (the thermal protection system) on the Space Shuttle, thus saving humans from a laborious task tha begins within minutes after hte Shuttle lands and ends just prior to launch - typically three to four months. Upon landing at either the Dryden facility in California or Kennedy Space Center in Florida, the orbiter is brought to either hte Mate-Demate Device (MDD) or the Orbiter Processing Facility (OPF). These large strucutres provide access to all areas of the orbiters.

The Shuttle is covered with several types fo heat resistant tiles that protect the orbiter's aluminum skin during the heat of reentry. While the majority of the upper surfaces are covered with flexible insulation blankets, the lower surfaces are covered with silica tiles. These tiles have a glazed coating over soft and highly porous silica fibers. The tiles are 95% air by volume which makes them extremely light but also makes them capable of absorbing a tremendous amount of water. Water in the tiles causes a substantial weight problem that can adversely affect launch and orbit capabilites for the shuttles. Because the orbiters may be exposed to rain during transport and on the launch pad, the tiles must be waterproofed. This task is accomplished through the use of a specialized hydrophobic chemical, DMES, which is injected into each and every tile . There are approximately 17,000 lower surface tiles covering an area that is roughly 25m x 40m.

In the current process, DMES is injected into a small hole in each tile by a handheld tool thpumps a small quantity of chemical into the nozzle. The nozzle is held against the tile and the chemical is forced through the tile by a pressurized nitrogen purge for several seconds. The nozzle diameter is about 1 cm but the hole in the tile surface is about 0.1 cm. The heights range from 290 cm to 400 cm from the floor of the OPF. It takes about 240 person hours to rewaterproof the tiles on an orbiter. Because the chemical is toxic, human workers have to wear heavy suits and respirators whie injecting the chemical and, at the same time, maneuvering in a crowded work area. One goal for using a robot to perform this task is to eliminate a very tedious, uncomfortable, and potentially hazardous human activity.

The tiles must also be inspected. By inspecting the tiles more accurately than the human eye, it is hoped that the Thermal Tile Servicing System will reduce the need for multiple inspections. During launch, reentry, and transport, a number of defects can occur on the tiles. These defects are evidenced as scratches, cracks, gouges, discoloring, and erosion of surfaces. The tiles are examined for such defects to determine if they warrant replacement, repair or no action. The typical procedure involves visual inspection of each tile to see if there is any damage and then assessment and categorization of the defects according to detailed checklists. Later, work orders are issued for repair of individual tiles.

The TTPS has three main parts: a mobile robot (called the Tessellator), a separate (off-board) computer (called the Workcell Controller) that controls the overall thermal tile processing tasks, and a human operator to monitor and control the other two components.

The Tessellator robot is designed to inspect each tile and inject the waterprooing chemical. Because there are so many tiles, Tessellator divides (or tessellates) its work area into uniform work spaces, inspecting tiles in each area with as little overlap between work spaces as possible.

Before each inspection shift, the operator enters instructions into the Workcell Controller about shuttle position and inspection sequence. The Workcell Controller workstation creates jobs for the Tessellator robot and updates other NASA databases after the robot uploads data gathered during the course of the shift. This data includes tile images, records of tiles injected and inspected, and other pertinent job data. In addition, robot status data is used to monitor robot operation.

At the beginning of the shift, the Tessellator is downloaded a job. The job consists of a series of files describing the locations, sequences, target IDs, orbiter parking measurements, etc. Tessellator then uses a rotating laser to position itself under the shuttle, and the robot's camera locates the exact tile to be inspected. Because the shuttle's belly is not flat, Tessellator customizes its upward movement to each tile:

Two vertical beams on either side of the robot raise the manipulator arm, which holds hte injection tools and camera. A smaller lifting device raises the arm the rest of the way.

By comparing the current state of each tile with the state of the tile at previous inspections, Tessellator characterizes anomalies in tiles as cracks, scratches, gouges, discoloring, or erosion. The robot also indicates when it is unsure of what is wrong with a tile, so the supervisor can reanalyze the tile on the screen of the Workcell Controller. At the end of a shift, Tessellator's updated tile information is entered into existing NASA databases.

On board, a computer controls Tessellator's high-level processing tasks while low-level controllers and amplifiers direct arm and wheel motions. Two more computers control the robot's vision and injection systems. If anything goes wrong - rising compartment temperatures, low battery level, or other changes - safety circuits will shut the robot down, and Tessellator will correct the problem.

MAPS(the mobility and positioning system) issues movement commands to the motor controller, which controls the mobile base of the robot. MAPS in turn is controlled either by the operator or an on-board computer called the Planner. The operator controls robot movement and positioning using a hand-held joystick. The planner controls robot movement and positioning by providing MAPS with a specification of the destination and route.

The Tessellator robot is unstable when the maipulator arm is extended, so stabilizer legs are used to provide stability. These legs must be retracted when the robot is in motion. MAPS is responsible for controlling the stabilizer legs.

Historical Information

We know of no previous robots used to service the orbiter thermal protection system nor of any attempts to build such a robot. Although the CMU Tessellator robot was delivered to NASA in 1994, as far as we know it has not been used in Shuttle operations. We have changed the design from the original in order to enhance safety and to make it a better example for our purposes.

Environment Description

MAPS is one subsytem of the Thermal Tile Processing System. MAPS stands for Mobility and Positioning Subsystem and is one of the subsytems functioning on the mobile robot itself. MAPS interacts directly with four other subsystems in the TTPS:

Tile Servicing Subsystem (TSS) - The TSS is responsible for directing and coordinating all tile servicing operations including positiong of the manipulator arm, operation of the vision subsystem and operation of the DMES injection system. The TSS is also responsible for directing movement of the mobile robot to the correct position for tile servicing.

Motor Controller (MC) - The MC is responsible for controlling the wheels and wheel motors on the mobile robot.

Aural and Visual Alert Subsystem (AS) - The AS is responsible for providing movement warnings to any humans in the area of the mobile robot during movement and operations.

System Log (SL) - The SL maintains a log of data and information recorded during TTPS operations for analysis and study of system performance and actions.

Assumptions

[EA.1] The MAPS subsystem is working within the TTPS.
Rationale: This intent specification describes the Mobility and Positioning Subsytem for the spa shuttle Thermal Tile Processing System. It is not intended for use outside of the TTPS project.

[EA.2] The mobile robot will be able to operate effectively in a crowded and busy workspac[MB-FR.2]
Rationale: The work areas of the Orbiter Processing Facility can be very crowded. The facilities provide access to all areas of the orbiters through the use of intricate platforms that are laced with plumbing, wiring, corridors, lifting devices, etc. After entering the facility, the orbiters are jacked up and leveled. Substantial structure then swings around and surrounds the orbiter at all sides and at all levels. With the exception of the jackstands that support the orbiters, the floorspace directly beneath the orbiter is initially clear but the surrounding structure can be very crowded. The robot must negotiate jackstands, columns, workstands, cables, and hoses. In addition, there are hanging cords, clamps, and hoses. Because the robot might cause damage to the ground obstacles, cable covers will be used for protection and the robot system must traverse these covers.

Constraints

[EC.1] The Tessellator Robot must be no more than 1.1m (42") wide when stowed.
Rationale: The facility personnel access doors are 1.1m (42") wide.

[EC.2] The Tessellator Robot must be no more than 2.5m (100") long.
Rationale: The layout within the OPI allows a length of 2.5m (100") for the robot.

[EC.3] The Tessellator Robot must be no more than 1.75m (70") high when stowed and extend to reach tiles between 2.9 and 4 meters away. [MB-FR.3]
Rationale: There are some structural beams whose heights are as low as 1.75m (70"), but once under the orbiter the tile heights range from about 2.9 meters to 4 meters.

[SC.1] Use of the TTPS must not cause or contribute to an unaccpetable loss (accident) adefined by Shuttle management.
Rationale: There is no gain in tile servicing or replacement if the automated system is not as safe as the existing system.

System Goals and High-Level Requirements

The System Goals and High-Level Requirements are described for the TTPS and broken down by subsystem. Only those goals and requirements relating to general TTPS function and the function of the MAPS subsytems are explained in detail, as the rest is out of the scope of this project. Note that the Safety-Related constraints are separated by subsystem in this specification for clarity, and are not all located in the Environment Description.

Thermal Tile Processing System (TTPS)

The Thermal Tile Processing System has components located in the control room and on a mobile rob. t These components operate together to achieve the system functional requirements.

System Goals

[TTPS-FG.1 The TTPS shall inspect the thermal tiles for damage caused during launch, reentry and transport. [TTPS-FR.1] [TTPS-FR.2]
Rationale: The purpose of the TTPS is to inspect thermal tiles on the space shuttle carefully ar thoroughly to detect damage and decide which tiles need replacing prior to the next flight of the particular orbiter being serviced.

[TTPS-FG.2 The TTPS shall apply waterproof chemicals to the thermal tiles. [TTPS-FR.3]
Rationale: The purpose of the TTPS is to utilize a robot to perform DMES injection into each ι the thermal tiles on the shuttle as the chemical is hazardous to human health.

High-Level Requirements

[TTPS-FR.1 The TTPS shall inspet each tile to identify tiles with defects.[TTPS-FG.1]
Rationale: Sensors on the TTPS will be able to detect damage unseen by the naked eye and better determine which tiles need to be replaced.

[TTPS-FR.2 The TTPS shall assess and categorize each defect identified. [TTPS-FG.1]
Rationale: The TTPS is not responsible itself for replacing damaged or defective tiles. It must create a log to indicate which tiles need replacement.

[TTPS-FR.3 The TTPS shall inject DMES into each tile. [TTPS-FG.2]
Rationale: The TTPS is responsible for correctly and accurately injecting DMES into each of the serviced tiles to waterproof the tiles.

System Limitations

[TTPS-L.1 Use of the TTPS must not negatively impact the flight schedules of the orbiters mothan that of the manual system being replaced.

[TTPS-L.2 Maintenance costs of the TTPS must not exceed TBD dollars per year.

Workcell Controller (WCC) Requirements and Constraints

The Workcell Controller is responsible for the overall processing being performed by the mobile robot.

Subsystem Goals

[WCC-FG.1 The WCC shall have control over the thermal tile processing performed by the mobile robot. [WCC-FR.1] [WCC-FR.2] [WCC-FR.3] [WCC-FR.4] [WCC-SC.1]
Rationale: The WCC is the hardware in the control room that has control over the Tessellator robot on the facility floor. Through this device, the system knows the job to be completed during any given shift and the opeartor can monitor the robot's operations.

High-Level Requirements

[WCC-FR.1 The WCC shall download data from the NASA databases for the Tessellator robot to use during a shift. [WCC-FG.1]
Rationale: Data will be stored reagarding what part of the orbiter has already been serviced. This information will be downloaded prior to beginning a shift to ensure no area is missed by the TTPS and no area is unnecessarily serviced twice.

[WCC-FR.2 The WCC shall create instructions for the Tessellator robot. [WCC-FG.1]
Rationale: The WCC will know the areas on the orbiter still to be serviced and can create the service schedule for the current shift.

[WCC-FR.3 The WCC shall monitor robot operation to ensure tasks are completed.[WCC-FG.1]
Rationale: The WCC is in the control room and is the human operator's link to the robot on the facility floor. Information passed through the WCC will allow human monitoring of robot operations.

[WCC-FR.4 At the end of a shift, the WCC shall update NASA databases with tile images, records otiles injected and inspected, and other pertinent data.[WCC-FG.1]
Rationale: The work databases must be updated with the work completed on each shift to ensure all areas of the orbiter are serviced and no area is unnecessarily serviced twice.

Safety-Related Constraints

[WCC-SC.1] The instructions provided to the Tessellator must not result in any tiles beinmissed in the inspection or waterproofong process. [WCC-FG.1] [H8]
Rationale: Per NASA safety plans, no tile shall go uninspected, therefor the WCC is responsible for issuing instructions for servicing areas that may overlap previously inspected areas, but cannot miss any tiles.

Mobile Robot Requirements and Constraints

The mobile robot consists of a mobile base carrying a tile servicing subsytem, a mobility and positioning subsystem, a location subsystem, a motor controller, a digital camera, a manipulator arm, a DMES injection subsystem, a vision subsystem, a system log, and everal safety subsystems included simply to provide safety functions including a safety fuse, a proximity-sensor, and an alerting subsystem.

Mobile Bas e (MB)

The Mobile Base (MB) provides the structure for the robot and all of its subsystems located in the orbiter processing facility.

Subsystem Goals

[MB-FG.1 The MB shall support, contain, and transport the tile servicing equipment[MB-FR.1] [MB-FR.2] [MB-FR.3]
Rationale: A stable and sturdy structure is necessary for the tile servicing equipment to function properly and within the specified accuracy limits.

High-Level Requirements

[MB-FR.1] The MB shall be able to carry all the mobile robot subsystem componen. [MB-FG.1]
Rationale: The MB is responsible for the robot structure and support of all subsytem components.

[MB-FR.2] The MB shall be able to move smoothly in any direction and to cross cablcovers on the floor. [EA.2] [MB-FG.1] [H3]
Rationale: The facility floor is very crowded with other workers and equipment. The MB needs to be able to smoothly pass over or manuever around obstacles during operation.

[MB-FR.3] The MB shall be able to raise its inspection and injection equipment to the levrequired for servicing the tiles, from 2.9 meters to 4 meters. [EC.3] [MB-FG.1]
Rationale: The robot must be able to enter the facility floor through a certain set of doors and under a beam, but then must extend to service the tiles on the shuttle.

Safety-Related Constraints

[MB-SC.1] The MB must protect against fire and explosion.[H6]
Rationale: In case of fire or other explosion in the facility work area, the MB needs to be protected avoid exposure of the DMES to humans in the area.

[MB-SC.2] The MB must be movable in case of an emergency.
Rationale: In case of an emergency on the facility floor or failure of the TTPS, the MB must be able to be moved by humans in the area to gain access to emergency exits.

Tile Servicing Subsytem (TSS)

The TSS coordinates the tile servicing activities of the Tessellator Robot, including positioning of the MB, positioning of the Manipulator Arm and DMES injection operations.

Subsystem Goals

[TSS-FG.1 The TSS shall direct and coordinate all tile servicing operations, including th positioning of the manipulation arm, the operation of the vision subsystem, and the operation of the DMES injection system. [TSS-FR.1]
Rationale: The TSS is the subsystem responsible for all positioning and tile servicing operations.

[TSS-FG.2 The TSS shall direct movement of the MB to the correct postion for tile servicing[TSS-FR.2]
Rationale: To facilitate the TTPS, the TSS will determine positioning and movement for the MB, as monitored by the human operator in the control room.

High-Level Requirements

[TSS-FR.1 The TSS shall plan the course of action required to complete a given tasl [TSS-FG.1]

[TSS-FR.1.1 The TSS shall raise the manipulator arm to the location of the tile to beserviced.

[TSS-FR.1.2 The TSS shall identify damaged tiles using the vision subsystem

[TSS-FR.1.2.1] The TSS shall compare the current state of each tile with the state of the tile at previous inspections.

[TSS-FR.1.2.2 The TSS shall characterize anomalies in tiles as cracks, scratches, gouges, discoloring, or erosion.

[TSS-FR.1.2.3 The TSS shall indicate if unsure what is wrong with a tile so the operator can reanalyze the tile on the screen of the workcell controller.

[TSS-FR.1.3 The TSS shall command the operation of the DMES injection subsystem

Rationale: The TSS shall be responsible for planning and commanding tile servicing opeartions to the other TTPS subsystems on the MB.

[TSS-FR.2 The planner shall determine the next work location and the most optimal route of travto it while avoiding obstacles[TSS-FG.2]

[TSS-FR.2.1 The TSS shall inform the operator about the next desired work zone location.

[TSS-FR.2.2 When the mobility and positioning system is being commanded by thon-board TSS, it shall provide the route of travel and a destination. Locations shall be provided in world coordinates. [MAPS-FG.2]

[TSS-FR.2.3 The TSS shall determine whether an adequate location has been achievedbefore beginning any tile servicing operations.

Rationale: For safety reasons, the operator is responsible for mobile basemovement. However, to allow for potential autonomy and thus more efficient operation, the TSS can directly issue movement commands to the mobility and positioning system. Any TSS movement commands must be monitored by the operator.

Safety-Related Constraints

[TSS-SC.1 The TSS must not move the manipulator arm unless the vision system is operation. [H4]
Rationale: The vision system is necessary for determining if it is safe to move the manipulator arm, the arm shall not move without visual confirmation.

[TSS-SC.2 The TSS must not command the manipulator arm into contact with any object unlesrequired for servicing. [H4]
Rationale: DMES chemical is toxic to humans, and could cause damage to other sensitive shuttle components. Manipulator arm contact must be minimized to avoid accidental exposure to DMES.

[TSS-SC.3 Chemicals must not be injected without providing a warning to humans in the are [H7]
Rationale: DMES is toxic to humans and alert must be issued in case humand are working in an area near to that of the MB.

[TSS-SC.4] The injection system must not be operated unless the mobile base is stopped in the work zone and the manipulator arm has moved the injection tool to the proper tile. [H7]
Rationale: Accidental release of DMES chemical must be avoided.

Mobility and Positioning (MAPS)

The MAPS subsytems is responsible for issuing movement commands to the MC based on the plans supplied by either the TSS or the human controller in the control room.

Subsystem Goals

[MAPS-FG.1] MAPS shall control the movement of the robot around the work area and position it in the appropriate locations in the hangar for tile servicing.[MAPS-FR.1]
Rationale: The MAPS subsystem is responsible for commanding robot movement in the work area.

[MAPS-FG.2] MAPS shall command robot navigation according to commands from an operator-controlled hand-held joystick or according to routes and destinations provided by the TSS[TSS-FR.2.2] [MAPS-FR.2] [MAPS-FR.3] [MAPS-FR.4]
Rationale: The MAPS subsystem accepts navigation commands either from the TSS when in computer controlled operations, or from the human operator in the control room via the joystick controller.

[MAPS-FG.3] MAPS shall control robot base stabilization and provide movement warnings.[MAPS-FR.5] [MAPS-FR.6]
Rationale: The MAPS subsystem will provide system warnings for movement to the humans in the work area and the human operator in the control room.

[MAPS-FG.4] MAPS shall store information about the operation of MAPS and the mobile robot as a whole in the System Log.[MAPS-FR.7]
Rationale: The recording of a variety of performance data will enable NASA system engineers to fine tune the operation of the robot and its software and also assist in maintenance and operational audits.

High-Level Requirements

[MAPS-FR.1] MAPS shall generate motor control commands to maneuver the robot to the zone required for the current job session. [MAPS-FG.1]
Rationale: MAPS is the subsystem responsible for issuing movement commands to the motor controller.

[MAPS-FR.2] MAPS shall provide a computer-controlled mode of operation where routes and destinations are provided by the on-board TSS. [MAPS-FG.2] [MAPS-SC.4]
Rationale. Human control of MAPS throughout the long and tedious tile servicing process is impractical. More efficient movement, particularly in tight spaces, can be commanded by the computer. Sufficient confidence cannot be obtained in the automated implementation of some safety-related robot operations, and therefore human movement control will be used during some limited but particularly hazardous TTPS operations.

[MAPS-FR.2.1] MAPS shall accept a set of target positions and final destination from the TSS

[MAPS-FR.2.2] MAPS shall generate appropriate commands to the motor controller to direct the MB to the final destination position via the immediate target positions.

[MAPS-FR.2.3] MAPS shall notify the TSS and the user interface when a satisfactory zone has been achieved or when it has failed to complete a commanded move and the reason for any failure.

[MAPS-FR.2.4] MAPS shall move into the Computer mode of operation in response to an appropriate message from the operator.

[MAPS-FR.2.5] MAPS shall notify the operator when it has entered Computer Mode.

[MAPS-FR.2.5.1] MAPS shall ignore all commands from the joystick while in Computer Mode.

Rationale: To prevent inadvertent joystick deflection from affecting robot movement, commands *from the joystick on the operator's control panel are ignored while in Computer Mode.*

[MAPS-FR.2.6] MAPS shal stop all motion and begin in Operator Mode upon receipt of a command to change from Computer Mode to Operator Mode.

Rationale: The robot must cease all motion when the command is issued to prevent hazardous movement of the robot.

[MAPS-FR.3] MAPS shall provide an operator-controlled mode of operation wher movement is commanded via a joystick located on the operator's control panel. [MAPS-FG.2] [MAPS-SC.4]
Rationale. Operator control is safer when environmental conditions are uncertain or in particularly hazardous state. Manual control will also be used during routine maintenance operations and at TTPS Powerup.

[MAPS-FR.3.1] MAPS shall default to Operator Mode at powerup or after any type c temporary shutdown or movement inhibition.

[MAPS-FR.3.2] In Operator Mode, movement commands to MAPS shal come from the position of the joystick on the operator's control panel.

[MAPS-FR.3.3] MAPS shall be able to respond to either single joystick motion commands c to any combination of commands.

[MAPS-FR.3.4] While in Operator Mode, all movement messages from the TSS shall b ignored.

[MAPS-FR.4] MAPS shall determine robot position using the Location Subsystem. [MAPS-FG.2]
Rationale: The TTPS Location Subsystem provides MAPS with MB position information in the work area.

[MAPS-FR.5] MAPS shall control the deployment and retraction of the stabilizer legs. [MAPS-FG.3]
Rationale: MAPS controls all robot movement, including the stabilization of the mobile base when it reaches its destination for tile servicing.

[MAPS-FR.6] MAPS shall control the activation and deactivation of the aural and visual alert syste.
[MAPS-FG.3]
Rationale: MAPS will be aware of all robot movement and the state of the mobile base at all times, therefore it is the subsystem to issue proximity warnings to humans in the work area.

[MAPS-FR.7] MAPS shall send messages to the system log about all events and errors MAPS sha send messages to the system log about all events and errors related to MAPS operation. [MAPS-FG.4]
Rationale: MAPS has complete information about the opearting state of the robot and shall send this information to the system log for later shift planning and system analysis.

Safety-Related Constraints

[MAPS-SC.1] Tolerances for movements must be modifiable after delivery.
Rationale: Movement space in the facility may change after the robot is in use.

[MAPS-SC.2] Accleration or decceleration when starting or stopping motion under norm circumstances must be low to allow a smooth start amd a smooth stop at the destination
Rationale. A smooth start and stop are necessary to minimize wear on the robot hardware and reduce maintenance time and cost. This is also a constraint that is a trade off with the need for the immediate stopping of movement of the robot in case of emergency.

[MAPS-SC.3] Mobile base acceleration, decceleration, and velocity must be modifiable after delivery.
Rationale. The appropriate units of acceleration and decceleration must be determine through trial and error. In addition, hardware changes in the robot or temporary or permanent operational changes in the OPF and the OPF environment could require changes to these values in the future.

[MAPS-SC.4] The mobile base must move only when commanded by the operator or whe commanded by theTSS and approved by the operator [MAPS-FR.2] [MAPS-FR.3]

[MAPS-SC.4.1] MAPS must not enter Operator Mode unless the joystick is physicallconnected to the robot and the joystick is in the neutral position.

[MAPS-SC.4.2] The robot must not move unless the deadman switch is depresse.d

[MAPS-SC.4.3] If the operator releases the deadman switch and then later depresseit again, all previous commands must be ignored and a new command must be issued before any robot movement occurs.

Rationale: A long enough time may exist between releasing the deadman switch and depressing it again that the environment may have changed and previous commands may no longer be safe.

[MAPS-SC.4.4] When the safety fuse is in the HALT state, the mobile base must not be capable of performing any kind of movement.

Rationale: All robot movement must stop if there is any doubt about a problem in one of the TTPS subsystems and the safety fuse was tripped.

[MAPS-SC.4.4.1] MAPS must not begin movement if the safety fuse is in the HALT state or if the state of the safety fuse is unknown.

[MAPS-SC.4.4.2] MAPS must stop all movement and notify the operator if the safefuse goes into the Halt state during a move or if the state of the safety fuse is indeterminable.

[MAPS-SC.4.5] MAPS must reinitialize itself and all the subsystems it controls when the safety fuse changes from HALT to SAFE state. Any previous uncompleted movement commands must be discarded.

Rationale: The system may be nonoperational for a long time while the problem thatriggered the safety fuse is being identified and fixed. When the system is restarted, the environment around the body may have changed and the previous movement commands may be inappropriate.

[MAPS-SC.5] The mobile base must stop when commanded.

[MAPS-SC.5.1] Release of the deadman's switch must cause the robot to cease motion.Motion must remain disabled until the switch is depressed again.

[MAPS-SC.5.2] When the operator releases the deadman switch, the robot must deccelerate quickly to avoid rolling into the obstacle the operator is trying to avoid.

Rationale. The operator is tasked to release the deadman switch during a move when the robot is about to impact an obstacle. Rapid decceleration is required to avoid rolling into the obstacle the operator is trying the avoid. However, such rapid decceleration is hard on the robot mechanisms and should be used only when necessary to avoid a hazardous condition.

[MAPS-SC.5.3] If the robot is in Operator Mode and the joystick is returned to the normposition, all robot motion must cease.

[MAPS-SC.6] The mobile base must not be commanded to an occupied position.

Rationale: Commanding the robot to an occupied position violates the system safety plan and the minimum separation between obstacles constraint.

[MAPS-SC.6.1] The operator must be kept informed of the location of the robot and obstacles in the work area.

[MAPS-SC.6.2] Human override capability must be maintained at all times in either operating mode.

[MAPS-SC.6.3] When operating in Computer Mode, MAPS must notify the user interface the direction and route of a move amd must require operator permission before commencing motion.

[MAPS-SC.7] The manipulator arm must move only when the stabilizers are fully extended.

[MAPS-SC.7.1] MAPS must ensure that the stabilizers are fully extended prior to enabling manipulator arm movement.

[MAPS-SC.8] The mobile base must not move when the stabilizers are extended.

[MAPS-SC.8.1] The stabilizers must be retracted prior to commencing motion.

[MAPS-SC.8.2] If stabilizer retraction or deployment fails, MAPS must notify the Operator athe TSS of the failure.

[MAPS-SC.8.3] Wheel motors must be turned off while the stabilizer legs are extended and powered back up when the legs are retracted.

[MAPS-SC.9] The manipulator arm must be stowed during all mobile base movement.

[MAPS-SC.10] The stabilizer legs must be deployed whenever the manipulator arm is not stowed.

[MAPS-SC.10.1] The manipulator arm must not be extended when the stabilizers are retracted.

[MAPS-SC.10.2] The stabilizers must not be retracted until the stabilizer arms are fully stowed.

[MAPS-SC.11] The mobile base must not move if any safety-related subsystems are nonoperational.

[MAPS-SC.11.1] MAPS must check that all safety-related systems are operational before beginning any movement.

[MAPS-SC.12] MAPS must provide movement warnings to workers in the facility floor.

[MAPS-SC.12.1] Visual movement alerts must start 10 seconds before mobile base movement begins and aural alerts must start 5 seconds before movement begins.

[MAPS-SC.12.2] Both visual and aural alerts must be activated continuously until movement is completed.

Rationale. The aural and visual alert system is provided to prevent injury to any human in the area. Because of the relatively low acceleration and velocity of the robot, five seconds should be adequate to allow humans to move out of the way. Alerts that begin too long before robot movement may lead to human delay in moving out of the way. The longer period for the visual alert is provided to allow humans to complete any critical actions before moving. These times may need to be changed on the basis of operational experience.

[MAPS-SC.13] The mobile base must not move while the DMES system is in operation.

Location System (LS)

The LS is responsible for providing information to MAPS regarding the robot location in the work area for robot movement.

Subsystem Goals

[LS-FG.1 The LS shall provide information about the location of the mobile base in the processing facility. [LS-FR.1]

Rationale: MAPS requires information about the exact location of the robot in the work area to determine movement to the next tile servicing area.

High-Level Requirements

[LS-FR.1] Upon request the LS shall provide the location of the MB in world coordinates with an accuracy of ± 10 centimeters. [LS-FG.1]

Motor Controller (MC)

The MC controls the wheels and wheel motors on the mobile base as instructed to by MAPS for MB movement.

Subsystem Goals

[MC-FG.1 The MC shall command the wheels and wheel motors on the MB when instructed to by MAPS. [MC-FR.1] [MC-FR.2]

Rationale: MAPS commands all movement of the MB and will issues motor on and off commands to the MC as necessary for movement.

High-Level Requirements

[MC-FR.1] The MC shall provide power to the motor that drives the robot whee. [MC-FG.1]
Rationale: The MC will turn on the appropriate whell motors for robot movement when commanded by MAPS.

[MC-FR.2] The MC shall provide modes of operation appropriate both for control by a automatecsystem and for control by a human operator. [MC-FG.1]
Rationale: The MC needs to be able to accept commands from MAPS based on TSS directions and human operator commands.

Safety-Related Constraints

[MC-SC.1] The MC must be able to stop the motion of the MB within 0.seconds of receiving a stop command.
Rationale: A command to stop the robot must be acted on immediately to avoid collision or othhazardous situations.

[MC-SC.2] The maximum velocity of the robot must be no more that 30cm/sec.
Rationale: This maximum velocity allows quick and safe stopping of the robot in case of emergency.

Digital Camera (DC)

The DC is used to detect obstacles in the path of the robot while the TTPS is operating in the work area.

Subsystem Goals

[DC-FG.1 The DC shall provide information to the operator about obstacles in the path of the Mduring movement.
Rationale: The DC is the obstacle alerting subsystem for the robot. It gives information to other subsystems regarding obstacles in the path of robot movement.

Manipulator Arm (MA)

The MA is the subsystem responsible for lifting the tile servicing tools and DMES injection equipment to the tiles being serviced. It is commanded byb the TSS.

Subsystem Goals

[MA-FG.1] The MA shall raise the inspection system and injection system components to the levof the tiles being serviced. [MA-FR.1]
Rationale: The MA is the structural base for the tile servicing and DMES injecting equipment and must raise to the tiles it is servicing on the orbiter.

High-Level Requirements

[MA-FR.1] The MA system shall provide the mobility necessary for the inspection ar DMES injection tools to reach the tiles on the orbiter. [MA-FG.1]
Rationale: The MA needs to rise from its stowed position for robot movement and robot entry onto the facility floor to the level of the tiles being serviced on the orbiter.

[MA-FR.2] The manipulator arm controller shall provide the position and status of the arm upcrequest directly to theTSS, the operator, or MAPS. [MA-FG.1]
Rationale. In the original CMU design, the TSS received information abouthe manipulator arm and was responsible for determining whether movement was allowed. However, because of the importance of the information for safety and the difficulty of verifying AI software, the system design was changed so that the information can be obtained directly by MAPS.

Safety-Related Constraints

[MA-SC.1] The manipulator arm must be designed to bemanually operated should the need arise.

[MA-SC.2] The manipulator arm design must keep the inspection and injection tools steady enouc to allow accurate operation.

[MA-SC.3] Movement of the manipulator arm must be capable of being disabled by the operator the planner.

DMES Injection Subsystem (IS)

The IS is responsible for applying the DMES chemical to the tiles being serviced by the TTPS.

Subsystem Goals

[IS-FG.1] The IS shall inject DMES checmical to the tile being serviced by the TTPS. [IS-FR.1] [IS-FR.2]
Rationale: The IS is the subsystem responsible for DMES injection into the thermal tiles.

High-Level Requirements

[IS-FR.1] The injection system shall be controlled entirely by the planner. [IS-FG.1]
Rationale: Only the planner knows when the MA is steady enough for DMES injection and when the robot has come to a complete and full stop for injection to occur to minimize any possibility of human exposure to the chemical.

[IS-FR.2] The injection tool shsall release DMES into a tile. [IS-FG.1]

Safety-Related Constraints

[IS-SC.1] DMES injection subsystem must be inhibited when the manipulator arm is stowed or motion.
Rationale: If the arm is not properly aligned with a tile being serviced, the DMES chemical could leak out and come into contact with other humans in the work area.

Vision Subsystem (VS)

The VS is responsible for analyzing the tiles and detecting damage that may call for tile replacement.

Subsystem Goals

[VS-FG.1] The VS shall perform the tile registration and inspection tasks.
Rationale: THe VS is the subsystem located on the MA that will be raised to each tile for inspection and injection.

High-Level Requirements

This is beyond the scope of this project. THe VS requirements would be documented here regarding sensors and reading taking and results.

System Log (SL)

The SL acts as the recorded data from each shift of operation of the TTPS. All data regarding robot movements and any failures is logged for later study and shift planning.

Subsystem Goals

[SL-FG.1] The SL shall provide an automated data recording and information transfer function to the

TTPS. [SL-FR.1] [SL-FR.2] [SL-FR.3]
Rationale. This automated function is expected to increase the data integrit completeness, anc accuracy of reports and thus to provide great values in tracking and planning work. Robot status information should be helpful in monitoring robot operatio.n

High-Level Requirements

[SL-FR.1: The SL shall be capable of holding the information collected during a work sh. [SL-FG.1]
Rationale: All data should be stored on the robot until it can be downloaded after a shift has been completed.

[SL-FR.2: The SL shall contain the information determined during system design to be useful for operations, maintenance, and safety or operations audits, including at least tile images, records of tiles injected or inspected, and robot status information. [SL-FG.1]
Rationale: All data that needs to be analyzed or is desired by system engineers needs to be cited for logging to ensure it is captured.

[SL-FR.3: Upon request, the SL shall transfer the information collected during a work shift tthe WCC. [SL-FG.1]
Rationale: In case the robot shutsdown due to a failure or emergency, the data from the SL may be helpful in determining the problem.

Safety Fuse (SF)

The SF acts as an emergnecy stop system for the TTPS. It can be triggered by a subsystem failure automatically, or by the human operator from the control room.

Subsystem Goals

[SF-FG.1] The SF shall provide an emergency stop function to the TTPS. [SF-FR.1] [SF-FR.2] [SF-FR.3] [SF-FR.4] [SF-FR.5]
Rationale: The SF functions as the emergency stop for the TTPS and can be triggered either by a subsystem or by the human operator from the control room.

High-Level Requirements

[SF-FR.1 The emergency stopping of the mobile base or the manipulator arm motion shall bperformed at a low hardware level via safety circuits. [SF-FG.1]
Rationale: A simple mechanisms is desired for the SF so that it is reliable for shutdown cases.

[SF-FR.2 If the SF detects as unsafe state, all robot motion shall be electrically inhibited within 0.1 seconds by stopping the operation of any hardware actuator on the mobile base, including those controlling the wheels, the manipulator arm, and the injection system. [SF-FG.1]
Rationale: To avoid a hazardous condition, all robot movement and processes must cease if the safety fuse is tripped to avoid an accident.

[SF-FR.3 Upon a status request, the safety shall indicate whether the robot is in a state where it cabe moved safely or not. [SF-FG.1]
Rationale: Additional information about the robot state is necessary to determine if the robot can resume operation after the SF has been triggered.

[SF-FR.4 Only the operator shall be able to reset the safety fuse. [SF-FG.1]
Rationale: This feature was changed from the original CMU Tessellator design. Providin the software, particularly the TSS software which is written using AI techniques, with this function will involve a safety analysis that would be at best expensive and at worst impossible. The triggering of the SF indicates a

serious condition that could lead to robot or orbiter damage and requires a high level of assurance that the condition has been removed before enabling movement again.

[SF-FR.5 The operator shall be able to query the SF for the cause of the shut-dow. [SF-FG.1]
Rationale: TO ensure the opeartor understands the cause for shut-down, or can determine the cause, the opeartor will be able to find out from the SF what subsystem caused the trigger.

Proximity Sensing System (PSS)

The PSS is responsible for detecting nearby obstacles during robot shift work and signaling the system about the proximity of potential collisions.

Subsystem Goals

[PSS-FG.1] The PSS shall protect flight hardware by providing proximity sensing to the TTPS. [PSS-FR.1]
Rationale: The PSS is the proximity alert system for the TTPS to avoid collisions with other obstacles in the facility work area.

High-Level Requirements

[PSS-FR.1 The contact strips in the Proximity Sensing System shall send a signal to the S in the event of contact with any external object. [PSS-FG.1]
Rationale. The SF can stop the base faster that a human operator can detect a signaand react. However, this system design decision has important ramifications with respect to operator complacency and alertness. Training and operational audits should be used to ensure that operators are not overly depending in the PSS to avoid accidents.

Aural and Visual Alert System (AS)

The AS is the subsystem responsible for alerting humans in the work area of the robot about robot movement to prevent accidents.

Subsystem Goals

[AS-FG.1 The AS shall provide warnings about mobile base movement to any humans in the area of the robot. [AS-FR.1] [AS-FR.2]
Rationale: To avoid obstacle collisions, the AS will warn humans in the area when the robot is moving so that they can be sure not to get in the way of the robot.

High-Level Requirements

[AS-FR.1] Whenever any part of the robot is in motion, aural and visual indications shall be provided to alert humans in the vicinity of robot movement. [AS-FG.1]
Rationale: Both aural and visual alerts are needed to account for any individual human visual aural deficiencies. In addition, visual warnings may be provided earlier than the aural ones in order to provide longer advance warning before more disruptive aural signals.

[AS-FR.2] The MB movement alert system shall be controlled by MAP. [AS-FG.1]
Rationale. As MAPS provides all commands to the motor controller, it knows whenever thbase is about to start movement and when it has stopped.

Operator Task Requirements

This section contains assumptions, requirements, and constraints involving operator tasks and behavior. The information is used in the operator task analyses, the design of the operator interface, the MAPS logic, operator(user) manuals, and training plans and programs.

[OP-FR.1] The operator shall supervise all robot base movement and tile servicing.
Rationale: If the operators are required to interact every few minutes with the system in order to monitor base moves, then the attractiveness of the system to users is far less that one that needs only infrequent attention. Therefore, the size of the work areas will be adjusted to satisfy the goal of approximately one base move per half hour. Once per half hour translates roughly into 80 moves during the course of rewaterproofing the orbiter, which results in a workspace of 300 tiles. In addition, with approximately 15,000 tile servicing steps and only a few hundred base moves at most, total task time is affeceted primarily by time to service a tile and very little by the time to move the MB.

[OP-FR.2] The operator shall authorize all robot base movements before they are performed including movements commanded by the TSS.
Rationale: The TSS can drive the robot more smoothly and directly then the operator, but the operator is required to oversee the robot movement both to monitor the TSS movement for errors and for safety assurance. With the over-ride control, the operator can stop the motion at any time and observe progress of the robot in the course of a move. This design feature was included in the original Tessellator robot because it allowed a simple to implement upgrade path for the software for robot autonomy. The operator-override option allows full autonomy mode but also allows the robot to be shut down at any time by the operator.

[OP-FR.3] The operator shall stop the robot immediately if any obstacle appears in the robot path.
Rationale: The operators will have adequate visibility of the work area to prevent collisions that may be undetected by other subsystems in the TTPS.

[OP-FR.4] The operator shall be responsible for handling SF alerts and for resetting the SF when the system is ready for restart.
Rationale: By requiring the operator to initiate the restart sequence, this ensures that the operator is aware that something caused a system reset and the SF can be queried or the reason for the reset can be investigated.

[OP-FR.5] The operator shall be able to drive the robot independently of the TSS by use of joystick.
Rationale: While the computer can move the robot precisely, human robot movement control may be safer in situations where the environment is uncertain. In addition, this design feature will be useful if during operations it is determined that simply monitoring robot movement leads to inadequate alertness on the part of the operator and more direct control is necessary to assure safety.

Controls
The Controls section describes the control panel for the human opeartor involved in robot operations.

Subsystem Goals
[Con-FG.1] The Controls shall allow effective achievement of all assigned operator tasks and responsibilities as determined by the task analysis and system safety analysis. [Con-FR.1] [Con-FR.2] [Con-FR.3] [Con-FR.4] [Con-FR.5] [Con-FR.6] [Con-FR.7]
Rationale: The Controls system is developed along with the operators to ensure that the proper information is communicated to the human opeartor for nominal system operations.

High-Level Requirements
[Con-FR.1] Controls shall include at minimum a joystick and a deadman switch. Additional controls, such as dials, switches, buttons, or keyboard shall be provided as deemed necessary to perform the tasks identified in the operator task analysis. [Con-FG.1]

Rationale: Too many input or output devices could cloud the operator's understanding of the system or cause accidental movement of the robot.

[Con-FR.2 A joystick shall be provided to allow the operator to control the movement of the robot base. [Con-FG.1]
Rationale: A joystick is required for human operator directed movement during Operator Mode.

[Con-FR.3 The operator must be able to provide fine-grain enough movement of the joystick to control robot motion accurately enough to avoid obstacles and damage to the Shuttle or the robot. [Con-FG.1]
Rationale: During Opeartor Mode control of the robot, all movement directions come from the operator via tha joystick.

[Con-FR.4 The joystick shall be capable of being effectively operated by a man or a woman with average manual dexterity and strength as defined in the IEEE Standard. [Con-FG.1]
Rationale: Any qualified opeartor should be able to easily and effectively use the joystick system.

[Con-FR.5 The operator shall be provided with a deadman switch that allows or inhibits robomovement. Movement must stop within 0.5 seconds after releasing the deadman switch. [Con-FG.1]
Rationale: The deadman switch will be used as sthe primary means for the operator to authorize immediate stopping of all robot movement.

[Con-FR.6 The operator shall be provided with an emergency stop facility that stops all robot motion, including the MA and IS, within 0.5 seconds of activating it. [Con-FG.1]
Rationale: While the deadman switch will be used to stop movement of the MB, thoperator needs the ability to stop other moving parts on the robot in case of emergency. The emergency stop button, which will be directly connected to the SF, also provides a backup to the deadman switch (which is implemented through software) in case the robot does not stop when the deadman switch is released because of a hardware failure.

[Con-FR.7 Upon request, the joystick controller shall initiate a joystick calibratio. [Con-FG.1]
Rationale: Joystick position must be calibrated to the feel of the current opeartor to ensure the expected movement of the robot occurs when the joystick is used.

Displays
The Display system deals with the presentation of information to the operator from the robot.

[Disp-FR.1 The GUI and other control panel displays shall provide the operator with enough information about the status of the robot and the work area that the operator is able to avoid hazards and to perform necessary operational tasks as determined by the operator task analysis.

Hazard List and Hazard Log

Accident Definition
An accident is an unacceptable loss, as defined by NASA Shuttle program management. Unacceptable losses and their severity levels are:

Level 1:
A1-1: Loss of orbiter and crew(e.g., inadequate thermal protection)
A1-2: Loss of life or serious injury in processing facility

Level 2:
A2-1: Damage to orbiter or to objects in the processing facility that results in the delay of a launch and/or

result in a loss of greater than TBD dollars.

A2-2: Injury to humans requiring hospitalization or medical attention and leading to long-term or permanent physical effects.

Level 3:
A3-1: Minor human injury (does not require medical attention or requires only minimal intervention and does not lead to long-term or permanent physical effects).

A3-2: Damage to orbiter that does not delay launch and results in a loss of less than TBD dollars.

A3-3: Damage to objects in the processing facility (both on the floor or suspended) that does not result in delay of a launch nor loss of greater than TBD dollars.

A3-4: Damage to Tessellator robot

Safety Policy

All hazards related to human injury or damage to the orbiter must be eliminateor mitigated by the system design. A reasonable effort must be made to eliminate or mitigate hazards resulting at most in damage to the robot or objects in the work area. For any hazards that cannot be eliminated, the hazard analysis as well as the design features and develoment procedures, including any tradeoff studies, used to reduce the hazard level must be documented and presented to the customer for acceptance.

Hazard level will be determined by worst potential severity. Hazards that can result in human injuor damage to the orbiter must be eliminated or mitigated if they are not judged to be physically impossible or they are caused by *physical* conditions that are judged to have a likelihood occurrence of more than one in a million over a 20 year period. All types of software (logical) errors will be considered to be possible and likelihood arguments cannot be used to reduce safety efforts related to those errors. A qualitative evaluation of software-related hazard likelihood is acceptable, but as with quantitative evaluations, must be justified to Shuttle Program management and cannot be based simply on the use of testing and good software engineering processes.

Hazard Log

The following hazards have been identified for the mobile robot. Only those hazards related to the operation of MAPS have been evaluated in detail for this example intent specification. A complete system hazard analysis would require full analysis of all the hazards.

H1: *Violation of minimum separation between mobile base and objects (including orbiter and humans.)*

Subsystem: MAPS, VS, PSS, MC, LS, AS, operator displays and controls
Operation/Phase: movement from one work zone to another
High-Level Casual Factors:
Uncommanded or unintended motion;
Not stopping when commanded or not stopping fast enough;
Operator issues command that violates minimum separation between robot and object;
MB commanded to unsafe position by TSS;
Movement commanded when a proximity-sensing or other safety related hardware system is inoperable;
Object moves into robot path.
Level and Effect: A1-2
Safety Constraints:
MB must move only when commanded;

MB must stop when commanded;
MB must not be commanded to an occupied position;
The opertor shall supervise all robot base movement;
The operator shall authorize all robot base movements before they are performed, including movements commanded by the TSS;
The operator must have information about objects in robot path and the ability to stop the MB within 0.5 sec;
Movement warnings must be provided;
MB must not move if any safety-related subsystem is not operational
MC must be able to stop the motion of the MB within 0.2 seconcof receiving a stop command;
Maximum velocity of the robot must be no more than 30cm/sec;
PSS shall send a signal to the safety fuse in the event of contact witany external object;
It must be possible to move the MB out of the way manually in case of emergency.

Analyses Performed:
Actions Taken:
Status:
Verification:
Final Disposal (Closeout Status):
Responsible Engineer:
Remarks:

H2: *Robot movement with stabilizers extended*
Subsystem: MAPS, stabilizer legs
Operation/Phase: Movement from one work zone to another
High-Level Casual Factor:
Robot moves without retracting stabilizers.
Level and Effect: A3-4
Safety Constraints:
MB must not move if stabilizers are extended;
Stabilizers must be retracted during all MB movement.
Analyses Performed:
Actions Taken:
Status:
Verification:
Final Disposal (Closeout Status):
Responsible Engineer:
Remarks:

H3: *Robot base becomes unstable*
Subsystem: MAPS, stabilizers
Operation/Phase: All
High-Level Casual Factor:
Stabilizers not deployed while arm extended;
Stabilizers retracted while arm extended;
Robot falls over while crossing covers or other obstacles.
Level and Effect: A1-2
Safety Constraints:
MA must move only when stabilizers are fully deployed;
Stabilizer legs must not be retracted until MA is fully stowed
Analyses Performed:
Actions Taken:
Status:
Verification:

Final Disposal (Closeout Status):
Responsible Engineer:
Remarks:

H4: *Manipulator arm hits something*
Subsystem: Planner, MAPS, VS, MA, TSS, PSS
Operation/Phase: All
High-Level Casual Factors:
Arm commanded into an object;
MB moves without arm being completely stowed.
Level and Effect: A2-1, A2-2
Safety Constraints:
MA must be stowed before movement starts;
MB must ensure accuracy of 10cm for positioning and 1mm for tilservicing tasks;
TSS must not move the MA unless the vision system is operational;
TSS must not command the MA into contact with any object unless required fctile servicing;
PSS must send a signal to the SF in the event of contact of MA witany external object;
MA must keep the inspection and injection tools steady enough to allow accuraloperation;
Movement of the MA must be capable of being disabled by the operator or tMAPS.
Analyses Performed:
Actions Taken:
Status:
Verification:
Final Disposal (Closeout Status):
Responsible Engineer:
Remarks:

H5: *Damage to robot caused by robot component operation or failure*
Subsystem: All
Operation/Phase: All
High-Level Casual Factors:
Mobile robot operates with low oil level.
Level and Effect: A3-4
Safety Constraints:
When the SF is in the Halt state, the MB must not be capable of performing arkind of movement.
Analyses Performed:
Actions Taken:
Status:
Verification:
Final Disposal (Closeout Status):
Responsible Engineer:
Remarks:

H6: *Fire or explosion*
Subsystem: All
Operation/Phase: All
High-Level Casual Factors:
DMES achieves explosive mixture.
Level and Effect: A1-2
Safety Constraints:
MB design must protect against fire and explosion.
Analyses Performed:

Actions Taken:
Status:
Verification:
Final Disposal (Closeout Status):
Responsible Enginee:
Remarks:

H7: *Contact of human with DMES*
Subsystem: IS, TSS, VS
Operation/Phase:
High-Level Casual Factors:
DMES released in the wrong place.
Level and Effect: A2-2
Safety Constraints:
MB must not move while DMES system is in operation;
Chemicals must not be injected without providing a warning to humans in the area;
IS must not be operated unless the MB is stopped in the work zone anthe MA has moved the injection tool to the proper tile;
DMES subsystem operation must be inhibited when the MA is stowed or imotion.
Analyses Performed:
Actions Taken:
Status:
Verification:
Final Disposal (Closeout Status):
Responsible Engineer:
Remarks:

H8: *Inadequate thermal protection*
Subsystem: IS, TSS, VS, MAPS, WCC
Operation/Phase: All plus operation of Orbiter
High-Level Casual Factors:
Damaged tiles not detected;
DMES not applied correctly.
Level and Effect: A1-1
Safety Constraints:
The instructions provided to the mobile robot must not result in any tiles beirmissed in the inspection or waterproofing process.
Analyses Performed:
Actions Taken:
Status:
Verification:
Final Disposal (Closeout Status):
Responsible Engineer:
Remarks:

Verification and Validation

Review Procedure

A review board that is independent of the development team verifies levels 1, 2 and 3 of MAPS. Multiple iterations of this review process can be performed to help ensure that the analysis reflects the actual operation of the system. Suggestions for improvement are added as necessary and evaluated in the next iteration of the review process.

Participants
None currently for the TTPS project.

Results
None to date.

Level 2: System Design Principles
The material in this level of the intent specification has been completed only for MAPS and those interactions of other subsystems with MAPS. The specifications for the other subsystem functions is beyond the scope of this project.

TTPS System Interface Design

Tile Servicing Subsystem
TSS → MAPS

Maps-Move: Used to command robot movement. The parameters indicate a route that MAPS is tfollow (a set of waypoints), with the last point on the route being the desired work zone.

Maps-Disable Used to indicate to MAPS that movement is currently not legal. For example, thwould be used to indicate that DMES application has not been completed or that the manipulator arm is not ready to be stowed.

Maps-Enable: Used to indicate to MAPS that movement is currently legal. This message is used treverse the Maps-Disable message.

MAPS → TSS

Status-Message: Used to return the success or failure of the Maps Move command. A status message is also generated any time Computer Mode is entered or exited. The message will contain a code indicating the success or failure of the commands, the latest position of the robot, and the command to which the status message is responding.

Display

MAPS → Display

MAPS-operating-mode Sent whenever the MAPS operating mode changes. Contains the new operating mode.

Display-Position-Report Sent any time MAPS reads the scanner and determines a new position for the robot. The parameters represent the location of the robot (X,Y, and 0) with respect to world coordinates. This information can be displayed numerically or graphically.

Display-Request-Move-Permissior Sent prior to making any move. The route specified must be displayed to the operator, preferably in a graphical fashion. The set of waypoints contains X,Y,0 coordinates of each waypoint that makes up the route. A maximum of 20 waypoints can be provided. In addition to displaying the route for the operator's review, the following textual message should be displayed:

"Permission to move along the displayed route is requested. Please depress and hold the deadman's switch to athorize this move. You may lift the deadman's switch at any time to suspend this move."

Display-error: Sent any time an error condition is encountered.

Control Panel

Control Panel→ MAPS

Select-Operator-Mode: Used to switch MAPS into Operator Mode.

Select-Computer-Mode Used to switch MAPS into Computer Mode.

Disable-Operation Used to disable MAPS operation. The message will normally be used during maintenance, for example, if some type of maintenance or checking operations are performed on the joystick.

Enable-Operation Issued at powerup and after a Disable-Operation command.

Joystick

MAPS → Joystick

Joystick-init Initializes the hardware to read values from the joystick. Sent once at powerup.

Joystick → MAPS

Joystick-zero: Sent in response to a Joystick-init command. Records the current offsets as the zero X,Y, 0 position. This should be called once with the joystick in the home position (offsets may be hardwired later).

Joystick-status: Sent in response to a Joystick-init command. It contains the status of the joystick, either operational or not operational (i.e., unplugged).

Joystick-position Sent as a move command to MAPS whenever the joystick stops in a non-neutral position and the deadman switch is depressed or whenever the joystick returns to the neutral position. Includes the position of the joystick in the x,y,0 values, with a range limit of -128 to +127.

Joystick-Button1 Sent whenever joystick button 1 is depressed or released.

Joystick-Button2 Sent whenever the joystick button 2 is depressed or released.

Log Manager

MAPS → Log Manager

Log: Logs events identified by a "log code" into the log file.

Motor Controller

MAPS → Motor Controller

Reset: Resets the motion control board to its default (powerup) state. Should be called before arother actions are peformed with the board. Sets the acceleration and deceleration for all four motors to the same value. Also sets the maximum velocity (centimeters per second). The velocity is measured at one of the wheel contact points. Velocity must be in the range of 0 to 30 centimeters/second.

Move-velocity(X,Y,0): Used for velocity mode. When received the motor controller performs the kinematics on the body relative (x,y,0) velocity specified in (in/sec, in/sec, radians/sec), and sets the appropriate wheel velocities. This routine causes motion to occur.

Move-relative(X,Y,0): Used for position mode.When received the motor controller performs the kinematics on the body relative (x,y,0) desisred position specified in (inches, inches, radians) and then moves the mobile base using the acceleration and velocity values from set_acceleration and set_velocity, respectively.

Stop: Causes the vehicle to deccelerate to a complete stop. The motors remain on and servoing. In velocity mode, all velocities are set to zero.

Motors-off: Turns all four motors off to a freewheeling state. Any queued commands will be flushed.

Motors-on -velocity: Turns on all four motors and initializes them for velocity servoing mode. Initial velocities are set to zero.

Motors-on-position: Turns on all four motors and initializes them for position servoing. The position queue is flushed.

Motor Controller → MAPS

Motion-status: Sends the current motion status: (1) the movement mode, (2) the on/off status of each of the four wheel motors, (2) indication that an error occurred, and (3) completion of a movement. Sent whenever the status changes.

Stabilizers

MAPS → Stabilizers

Legs-Down: Causes the stiff legs to be deployed. If the stiff legs are already deployed, the message has no effect. In either case, the error-status parameter returns a success or failure code.

Legs-Up: Causes the stiff legs to be retracted.If the stiff legs are already retracted, the message has no effect. In either case, the error-status parameter returns a success or failure code.

Position-Request: Request for the stabilizer leg controller to provide the current status of the stabilizers.

Stabilizers → MAPS

Stabilizer-Status-Message: Sent in response to a legs-down, legs-up, or position-request message or whenever the status of the stabilizer changes.

Laser Scanner

MAPS → Scanner
Get-Scanner-Position Requests the latest position calculated by the scanner.

Scanner → MAPS
Send-Position: Provides the current position (x,y,0) in world coordinates. Sent in response to a Get-Scanner-Position message.

Safety Circuit

MAPS → Safety Circuit

Read-Safety-Circuit: Request for safety-circuit status.

Safety Circuit → MAPS

Safety-Circuit-Status: Sent in response to a Read-Safety-Circuit message or whenever the safety circuit status changes.

Arm Controller

MAPS → Arm Controller

Check-Arm-Positio: Request for arm status.

Enable-Arm-Movement Enables arm movement.

Disable-Arm-Movement Disables arm movement.

Arm Controller → MAPS

Arm-Status: Sent in response to any command from MAPS.

Motion Alert System

MAPS → Motion Alert System

Check-alert-system-status : Requests alert system status.

Activate-visual-alert: Sent to initiate a visual alert.

Deactivate-visual-alert: Sent to stop a visual alert.

Activate-aural-alert: Sent to initiate an aural alert.

Deactivate-aural-alert: Sent to stop an aural alert.

Motion Alert System → MAPS

Alert-system-status: Sent in response to a check-alert-status message or whenever the alert system status changes.

Controls and Displays

Control Panel

[DP.1] The control panel provides the operator with the ability to issue commands to t planner, to MAPS, and to other system components and to check their status.

[DP.1.1] Common action such as setting the Operator or Computer Mode ar implemented using buttons, dials, or switches rather than requiring more tedious and time-consuming keyboard inputs or even mouse clicks. Keyboard and mouse entries should be limited to infrequent activites or those that cannot be implemented using buttons, switches, or dials.

[DP.2] The control panel includes an emergency stop button that directly activates the S and shuts down power to all robot components within 0.5 seconds of being pushed. The emergency stop button must be located where it is always within the operator's reach.

[DP.3] A deadman switch is used to authorize or stop MB movement. Th deadman switch is activated by two joystick buttons, one on top and one on the bottom of the joystick handle. Movement does not occur unless one or both sticks are depressed and stops within 0.5 seconds after the button(s) is released.

[DP.4] The user interface can invalidate certain operator option when such operations are not legal or unsafe. For example, selection of Computer or Joystick mode is possible only when the SF is not in the HALT state.
Rationale. Although MAPS should ignore such illegal or unsafe commands, an extra lev of protection is provided by this design. Care must be taken not to block any operator options that might be needed by the operator, particularly in emergency.

Joystick

[DP.1] The joystick controller must be initialized prior at powerup. This involv setting the maximum velocity; x,y, and 0 thresholds, max throw constants, and speed factor.

[DP.2] The operator drives the robot by deflecting a joystick in the direction the operat would like the robot to travel. Deflection of the joystick away from the operator results in forward robot motion while deflection towards the operator results in backward robot motion. Deflection to the left and right produces corresponding robot motions to the left and right.

[DP.3] The robot is rotated by rotating the joystick handle. Rotation of the joystick han in a clockwise direction results in clockwise (as viewed from above) rotation of the robot. Likewise, counterclockwise rotation of the handle results in counterclockwise robot rotation.

[DP.4] Speed is controlled according to the amount of deflection of the joystick. Motion proportional such that a small deflection of the joystick results in a slow movement while a larger deflection results in a faster motion.

[DP.5] The joystick has a neutral position that signals the joystick is not commanding an motion. The joystick provides its position relative to this neutral position in the form of x,y, and 0.

Displays

The displays provide the following information: position of the robot, position of the stabilizer legs (deployed or not deployed), position of the MA (stowed or not stowed), the route provided by the planner, the status of the SF and the reason for being in the HALT state if it is, a view of the area ahead of and round the robot, and pictures of the tiles.

Operator Task Design Principles

Preliminary Task Analysis for MAPS

T1 Enter instructions for TSS into Workcell Controller
T2 Powerup and Initialize Tessellator
  T2.1 Initialize laser scanner and make any laser scanner ber code changes.
  T2.2 Set normal and emergency acceleration and deceleration values.
  T2.3 Calibrate joystick.
  T2.4 Set Operator Mode parameters (max velocity; X,Y,0 thresholds).
T3 Set or change MAPS operating mode.
  T3.1 Determine appropriate mode.
  T3.2 Select mode.
T4 Monitor Computer-Controlled Movement
  T4.1 Read displayed route
  T4.2 Authorize all mobile base movement
    T4.2.1 Check that stabilizers are retracted
    T4.2.2 Check that manipulator arm is stowed
      T4.2.3 Read and check dislayed route
    T4.2.4 Press deadman switch
  T4.3 Monitor movement
    T4.3.1 Monitor movement on screen
    T4.3.2 Release deadman switch id obstacles observed in path
  T4.4 Process error messages
T5 Control mobile base movement and positioning using the joystick
  T5.1 Check screen for obstacles
  T5.2 Check that stabilizers are retracted
  T5.3 Check that manipulator arm is stowed
  T5.4 Depress deadman switch
  T5.5 Operate joystick
  T5.6 Release deadman switch and return joystick to neutral position
  T5.7 Process error messages
T6 Check tile state on screen of Workcell Controller if TSS asks for help
T7 Monitor tile servicing
T8 Handle errors and failures
  T8.1 Handle safety fuse reset
    T8.1.1 Query fuse for cause
    T8.1.2 Take corrective action
    T8.1.3 Reset fuse after problem has been fixed
  T8.2 Process system error messages
  T8.3 Notify maintenance about breakdowns
  T8.4 Manually stow manipulator arm
  T8.5 Manually extend or retract stabilizer legs
  T8.6 Manually turn off wheel motors
  T8.7 Press emergency stop if unsafe conditions occur

Movement and Positioning Design Principles

MAPS Initialization

[DP.1] During system initialization, MAPS resets the MC interface.

[DP.2] During system initialization,MAPS initializes the joystick.

[DP.3] During system initialization, MAPS establishes that the robot iin a proper and safe startup state.
  [DP.3.1] MAPS verifies the SF is in the SAFE state.
  [DP.3.2] MAPS verifies the MB is stopped.
  [DP.3.3] MAPS verifies the LS and AS are operational.

[DP.4] During system initialization, MAPS determines the initial positio of the robot and sends a status message to the operator.

[DP.5] MAPS does not accept any non-initialization commands until initialization is complete.

MAPS Movement Control

[DP.1] MAPS issues movement commands only if the following conditions are true:
SF is in the SAFE state
MA is stowed
Stabilizer legs are retracted
Joystick is in the neutral position
Deadman switch is Depressed
SF and AS are both operational

[DP.2] MAPS issues visual and aural warnings to humans in the TTPS work area.
[DP.2.1] MAPS activates a visual alert 10 seconds before MB movement begins.
[DP.2.2] MAPS activates an aural alert 5 seconds before movement begins.
[DP.2.3] Both alerts are shutdown in Computer Mode when the final destination is reache or the deadman switch is released.
[DP.2.4] Both alerts are shut down in Operator Mode when the joystick is returned to a neutral position.
Rationale. The aural and visual alert system is provided to prevent any injurty to humans in t area. Because of the relatively low acceleration and velocity of the robot, five seconds should be adequate to allow humans to move out of the way. Alerts that begin too long before robot movement may lead to human delay in moving out of the way. The longer period for the visual alert is provided to allow humans to complete any critical actions before moving. These times may need to be changed on the basis of operational experience.

[DP.3] When informed by the TSS, the operator, or the SF, that motion is not legal, MAPS stops all movement operations and prohibits any further operations until informed that movement is again allowed. Once movement becomes legal again, MAPS will return to Initialize mode to reset all movement values.
Rationale. Starting up movement in the previously active mode is dangerous as different procedur could still be running in the robot processor, casuing the robot state to be different than the operator's assumed state of the system. This could cause serious system problems, thus a system reset in Initialize mode is necessary.

[DP.4] The default acceleration and deceleration values are normally used, bu acceleration and deceleration values can also be changed via the user interface. An emergency deceleration value is used for an emergency stop when the operator releases the deadman switch.

[DP.5] When commanding movement, translation in the x-y plane is performed firs followed by any required rotation. Either can be skipped if they are not necessary.

[DP.6] A movement is considered complete only whe the robot arrives in the desired work area.
[DP.6.1] In Computer Mode, the work zone is assumed to be the last waypoint on the route.
[DP.6.2] In Operator Mode, the robot is assumed to be in the work zone only whe the deadman swtich is released and the joystick returns to the neutral position. A new movement command must be issued for further movement to occur.

[DP.7] While external position information will be provided to MAPS in world coordinat, MAPS must issue commands in robot-relative coordinates. World coordinates must be translated to robot-relative coordinate outputs. MAPS uses a standard package of matrix math routines to do this converison.

Position Determination

[DP.1] Position determination occurs immediately following system initialization.

[DP.1.1] In Computer Mode this occurs prior to the beginning and completion of each route segment.

[DP.1.2] In Operator Mode this occurs when the joystick is returned to the neutral position.

Rationale. Position determination after a move is used to provide feedback on previous commands in order to detect errors in carrying them out. MAPS must also know the current robot position before it can send a new relative displacement command to the MC. It is unsafe to assume that the robot position is the same as after the last movement command.

[DP.2] Any time the robot position is determined, a message is sent to the user interface indicating the present location of the robot.

Stabalization

[DP.1] Prior to attempting movement, MAPS turns the drivetrain motors on, ensures that the arm is stowed, disbales the MA, and issues a command to retract the stabilizers.

[DP.2] Once the robot has arrived at the commanded work zone, MAPS commands deployment of the stabilizers and turns the motors off.

[DP.3] Once the stabilizers are fully extended, MAPS enables arm movement.

[DP.4] In the event stabilizer retraction or deployment fails, MAPS notifies the operator and the planner of the failure.

Rationale. The planner needs to know if stabilizer deployment has failed so it does not send any arm movement commands. The operator needs to be informed so the failure can be diagnosed and repaired.

Movement Control Mode Selection

At any time, MAPS is in one and only one of the following four modes: Initialization, Computer Mode, Operator Mode and Halted Mode.

[DP.1] Initialization Mode MAPS enters initialization mode at powerup and after SF reset.

[DP.2] Operator Mode MAPS transitions to Operator Mode from Initialization Mode, after the deadman switch is released, or upon receipt of a command from the Operator to change to Operator Mode. If a movment message is received from the TSS while operating in Operator Mode, that message will be ignored and an error message returned to the TSS.

[DP.3] Computer Mode MAPS changes to Computer Mode only upon receipt of a message to do so from the operator. If a joystick move command is received while operating in Computer Mode, that command is ignored and an error message returned to the operator.

Rationale: If the operator wants to change to Operator Mode while the TSS is generating movement commands, the fastest and best way to do this is to release the deadman switch. An accidental movement of the joystick should not result in a new Mode or movement command while in Computer Mode.

[DP.3.1] When MAPS receives a message to change to Operator Mode, all current waypoints and movement messages are discarded. Any subsequent computer-controlled movement messages require a new movement command from the TSS.

Rationale. The most likely reason for the operator to interrupt Computer Mode operation is that he or she has detected a problem with the route displayed, such as unsafe conditions along the route. In addition, the robot position or environment may have changed while in Computer Mode, and previous movement messages may be obsolete.

[DP.4] Halted Mode MAPS moves into Halted Mode whenever the system is in a state where movement is unsafe. This occurs when it detects that the SF is in the HALT state. MAPS transitions to Initialization Mode from Halted Mode when the SF returns to SAFE state.
Rationale. While the robot is not operational, subsystems or the environment may chang state from that assumed by MAPS or the operator. After leaving the Halted Mode, transition back to Computer or Operator mode will require reinitialization of these modes and updating any MB or environment state information used by MAPS.

Computer Mode

[DP.1] Initializatior Any time the operating mode changes to Computer Mode, before issuing any movement commands MAPS first selects position mode for the motor control interface, determines the position of the robot, and generates a status report to the TSS and operator.

[DP.2] Motor Control In Computer Mode, the motors are controlled using position moc. MAPS makes all requests to the MC using a move-relative command, which is provided with a value fo x and y or for 0. Preset acceleration, deceleration, and velocity values are used. The normal stop command is used to allow for a gradual stop when MAPS is completing a move while the emergency stop command is used for a rapid deceleration of the interruption of motion is requested from the operator through the deadman switch.

[DP.3] Route Determinatior The route to the work area as well as the final destination is specified in a message from the TSS. A route that contains zero length is logged as an error and a normal stop is commanded.

[DP.3.1] A "route" consists of a movement to zero or more intermediat locations, and then a single movement to a final terminal location.

[DP.3.2] If the desired movement consists of a single route segment, the move i conducted as a straight line from the current robot location to the desired work zone. If the move consists of more than one route segment, then the move is conducted as a sequence of straight line moves between the waypoints.

[DP.3.3] MAPS moves the robot along a straight line from the starting point relative t the robot, considered the origin, to a point specified in the move instructions. Given a x distance and a y distance, MAPS issues the appropriate commands to cause the robot to move a distance equal to $^2 + y^2$. If MAPS is given a rotation value for 0 degrees, MAPS instructs the robot to rotate 0 degrees relative to its starting position when the move command is given.

[DP.3.4] MAPS will need to calculate new coordiantes during a move any time th operator releases the deadman switch and the robot pauses. Given the values X', Y' and 0' read from the current position, new values X,Y and 0 are computed by:

new_X = Final _X - X'
new_Y = Final_Y - Y'
new_0 = Final_0 - 0'

[DP.3.5] If a move is calulated or commanded that results in a zero or near zero (les than 10 centimeters) length, the move is not executed.

Rationale: This requirement is used to prevent commanding a near zero lengt move and thus "jittering" around zreo. In addition, the limit of required accuracy of the LS is 10 centimeters.

[DP.4] Movement Contro When operationg in Computer Mode, MAPS acepts movemen commands from the TSS and issues motor commands to the MC to move the robot.

[DP.4.1] Before starting motion, MAPS notifies the user interface of the direction an route of a move and waits for operator permission. It does not begin the move until it detects that the deadman switch has changed from the released to the depressed position. MAPS then continues along all route segments as long as the switch remains depressed. If the deadman switch is released, MAPS stops all movement and reverts to Operator Mode.

Rationale. Releasing the deadman switch before the movement is complete is likely result from the operator having information about possible obstacles in the computer-generated path. The operator will need to take over at this point.

[DP.4.2] For each motion command, MAPS takes a reading of the current robot positior, calculates the difference between the actual robot location and the desired robot location, converts the difference into a robot relative move and performs the move. Following completion of the move, MAPS takes a reading of the robot location.

Rationale. As with any physical motion, there is a very real possibility that the actumotion does not exactly match the intended motion. Feedback from the outside environment must be taken into account to accomodate this potential error and determine if the position achieved is "close enough" to the ideal location by using a set of tolerances. The built-in tolerances can be changed during system initialization. If the desired location was not achieved, supplemental moves are calculated and commanded up to the maximum number of attempts allowed.

[DP.4.2.1] An attempt is a series of one or more commandecmoves, each preceeded and followed by a scanner query and a check if the target has been reached.

[DP.4.2.2] MAPS uses a set of coordinate tolerances forintermediate route locations and a set of tight coordinate tolerances for the completion of the final route location.

[DP.4.2.2.1] If the robot is moving via intermediate locations, a loosetolerance of 6 inches and 5 degrees is used for the inermediate loations. If after 10 attempts the robot position is still outside the tolerance limits, MAPS notifies the TSS and the user interface of the error and stops all motion.

Rationale. Ten attempts should be adequate to reach the desired position. Ten unsuccessful attempts probably indicates a problem that needs to be handled by the TSS or the operator.

[DP.4.2.2.2] Following the completion of the final route segmentthe current robot position and orientation are compared against the desired robot location and orientation. If a difference of more than 2 inches or 1 degree exists, up to 10 supplemental moves are calculated and attempted. If after 10 supplemental moves the robot position is still outside the tolerance limites, MAPS notifies the TSS and user interface of the error and stops all motion.

[DP.4.3] The robot continues to move until one of the following events occurs: the robot has arrived at the desired destination, the deadman switch is released, the SF leaves the SAFE state, or another error condition is detected.

[DP.5] Status Messages: MAPS sends the following status messages:

When Computer Mode is entered or exited, MAPS generates a status message to the TSS and user interface.

When MAPS completes a commanded move, the TSS and the user interface are notified.

If MAPS fails to complete a commanded move, an error message is returned to the TSS and the user interface identifying the reason for the failure.

When the final destination has been reached, MAPS notifies theTSS and the user interface.

If a movement message is received from the planner when not operating in Computer mode, MAPS returns an error message to the planner and to the operator.

If a joystick deflection occurs while in computer mode, MAPS informs the operator.

Operator Mode

[DP.1] Initialization When the operating mode changes to Operator mode and before issuing any movement commands, MAPS ensures that the joystick is connected and in the neutral position, selects velocity mode for the motor control interface, determines the position of the robot, and generates a status report to the TSS and operator.

[DP.2] Motor Control When Operating in Operator Mode, all move requests are made through a move-velocity command. A multiplier is used to increase or decrease the velocity in joystick mode. The values of the speed factor can be set from the interface and changed interactively during MAPS execution. Stopping motion is achieved by issuing a move-velocity request of zero for X,Y, and 0.

Rationale. Move velocity commands are more appropriate for joystick control and better match an operator's mental model of how movement will occur under joystick control. Being closer to what humans naturally expect will make control and monitoring more effective and efficient. It should also reduce training requirements, particularly when an emergency or high stress situation occurs.

[DP.3] Movement Control MAPS converts operator deflections of the joystick into movemercommands to the motors. Translation from world coordinates to robot relative coordinates is performed.

[DP.3.1] MAPS accepts the following values as parameters that can be reset durig operations:

-*Maximum velocity* The maximum velocity that can be commanded by the joystick in ardirection. The units used are inches per second for X and Y and radians per second for 0

-*X,Y, and 0 threshold*: These thresholds values give the minimum amount of joystick deflection required before that axis will be considered to have moved from the centered position. The values are determined during joystick calibration.

Rationale. The joystick will almost never return to the absolute location when released The nature of the mechanics are such that the "zero" position of the joystick is slightly variable. As a result, the "dead zone" that corresponds to an undeflected joystick must be expanded slightly to allow for this difference.

-*Joystick maximum throw constants* In order to provide a proportional joystick, the amount of joystick deflection relative to the full-scale deflection must be calculated. These maximum throw constants, one for each of X,Y, and 0, indicate the number that the joystick interface returns when the joystick if deflected full scale along the indicated axis.

-*Joystick speed factor* This value allows the tester or system integrator to select a "high gear" or "low gear" for the robot during run-time operations. The value is externally visible and can be set through the interface, but it is not intended to be an operator feature. Allmovement commands are multiplied by this value prior to being sent to the motor controller.

[DP.3.2] A joystick reading must be scaled into a percentage of the total deflectiopossible. The ratio of the current reading to the total possible deflection is first calculated. This percentage is then applied to the maximum possible velocity to yield a scaled velocity between zero and the maximum velocity, proportional to the reading supplied.

[DP.3.3] All robot mobile base motion ceases when the joystick is returned to thneutral position. Joystick deflections are ignored if the deadman switch is not depressed.

Information and Error Logging

MAPS records events (including both errors and succesful moves) intothe SL.
The following events are logged:

- A move or command is completed successfully.
- The safety fuse changes to the HALT state.
- The safety fuse changes to the SAFE state.
- MAPS enters Computer Mode.
- MAPS enters Operator MOde.
- A transition in to Computer Mode is interrupted and cancelled.
- A transition in to Operator Mode is interrupted and cancelled.
- MAPS operation is enabled.
- MAPS opertion is disabled.
- MAPS operation is stopped due to the operator releasing the deadman switch during a move.
- MAPS begins a movement along a segment other than the final segment in computer mode.
- MAPS begins movement along the final segment of a move.
- MAPS attempts a retry of the same move.
- The number of attempts tp reach a location exceeds the threshold limit.
- MAPS completes a route successfully.
- A motor control error is detected.
- The joystick is not connected.
- The joystick is deflected away from the neutral position, preventing MAPS from entering Operator

Mode.
- A motor control failure occurs during joystick operations.
- The initialization of the joystick fails.
- Retraction of the stiff legs fails.
- Deployment of the stiff legs fails.
- MAPS is unable to read the scanner data file during initialization.
- A position is read from the scanner.
- MAPS is unable to read scanner data.

Mobile Base

[DP.1] Tessellator's chassis and actuators are stiff.
Rationale: The Tessellator's high stiffness design ensures therobot's accuracy.

[DP.2] Omnidirectional wheels move the robot smoothly in any direction and over cable covers on tlfloor.

[DP.3] The MB is formed by a rigid welded steel frame in order to provide a very stiff base fromwhich to operate the manipulator. The base supports two enclosures for electronics and waterproofing equipment as well as an on-board nitrogen tank and a battery cage.

[DP.4] Automated screw jacks can be commanded to descend from the base and contacthe floor. Once the base reaches a particular work area, the stiff legs are deployed from the base to descend and contact the floor in order to provide a rigid base from which to work. Before the MB moves, the stiff legs must be retracted.
Rationale. Although the roller wheels on the mobile base are not soft and have a high durometerrating, they are compliant enough to affect accuracies and stability of the system at full MA reach. The screw jacks are needed to lock out this compliance and provide a stable, non-compliant platform. In addition, in order to reduce the amount of MB movement and to perform the required task in work zones that are not free of obstacles, the MA must sometimes reach out beyond the perimeter of the base and additional stability is required under these conditions.

[DP.5] The exposed parts of the MB are constructed of materials that are intrinsically safe

[DP.6] Electrical components are sealed in aluminum enclosures and purged with gaseous nitrogenHeat pipes cool the electrical compartments and the injection tool plate without circulating chemical-laden air through the electronic parts.
Rationale: DMES is flammable as well as toxic. E lectrical components need to be isolated from the chemical.

Location Subsystem

[DP.1] The LS provides the location of the robot in *world coordinates* (*x* and *y*) with respect to a given position in the orbiter facitlity. To determine position with the required accuracy, the LS uses a laser scanner and information from the VS.

[DP.2] A rotating eye laser scanner reads bar-code targets that are precisely located in the facility. Triangulation from three or more of the many bar code targets gives robot position within a few centimeters.

[DP.2.1] The laser scanner only takes position readings while the MB is immobile
Rationale. The scanner triangulations calculations assume a static base and cannotcorrect the readings to take into account the MB motion. If the positioning system changes, then this design principle may no longer hold.

[DP.2.2] The laser scanner must be initialized by the operator before first use with bacodes and scanner codes. Any changes to these locations after the initial setting will be made by the operator.

[DP.3] Position is determined using three frames of reference and corresponding transformbetween them. First, the orbiter is parked within some position error that is known and measured as a normal procedure. This position and position error data provides the ability to compute an orbiter-facility transform. Second, the transform between the robot and the facility is given by the laser positioning system. Third, the loop is closed through the vision system, which precisely identifies the position of a specific tile whose position on the orbiter is already known. Together, this information provides the ability to determine a precise robot-orbiter transform.
Rationale. Triangulation from three or moreof the many targets can give robot position within few centimeters, which is precise enough to find a specific tile. The orbiter tile positions are known very accurately and the tile position can be registered with the vision system used for inspection.

Motor Controller

[DP.1] Mecanum wheels use a novel roller design to obtain three-degree-of-freedom motion in thplane, pure rolling contact for accurate postioning, and non-singular motions for small and precise final motions.
Rationale: The size constraints of the vehicle coupled with the close quarter navigation needed for operating in the OPF requires a locomotion system of high maneuvarability.

[DP.2] The drivetrains for locomotion are in the diameter of the wheel hub and consist of a brushlesDC motor, resolver for positioning and communication, a brake, a cycloidal reducer providing 225:1 gear reduation with exceptional stiffness, and a locking hub that couples the output of the reducer to the wheel. The locking hub allows the operator to disengage the wheels directly from the drivetrain completely.
Rationale. In an emergency, the ability to disengage the wheels will allow towing or pushing t machine out of the way.

[DP.3] The drive system is able to move the robot over 10cm high steps and up tp 20% grades.

[DP.4] The drivetrain suspension is a simple rocker-arm design much like those on heavconstruction equipment. This design is very simple and acceptable for the top robot speed of 30cm/sec. The MB supports two enclosures for electronics and waterproofing equipment as well as an on board nitrogen tank and battery cage.

[DP.5] When commanded to do so, the MC provides power to the motor, which drivethe robot wheels. The MC accepts two kinds of commands. In position or relative displacement mode, the kinematics on the body relative (x,y,0) desired position are computed and the robot is driven in the desired direction based on previously set acceleration and velocity values. The second mode is velocity mode, where the kinematics on the body relative (x,y,0) velocity are computed and the appropriate wheel velocities are set.

[DP.6] All position (x and y) and rotation (0) information must be provided to the MC *robot-relative* coordinates.

[DP.7] The operator can set the normal and emergency acceleration and deceleration values cpreset defaults can be used. During a normal stop, the normal deceleration values are used. During an emergency stop, the robot will stop faster, using the emergency deceleration value.

[DP.8] The MC must be reset before any actions are performed and following any error.It takes approximately 3 seconds for the motion controller and the motion enabling hardware to properly configure.

Manipulator Arm

[DP.1] Because the shuttle belly is not flat, Tessellator customizes its upward movement to eactile: Two vertical beams on either side of the robot raise the MA, which holds the injection tools and camera. A smaller lifting device raises the arm the rest of the way.

[DP.2] A physical interlock is used that does not allow the MA to be deployed while thstiff legs are

retracted.
Rationale. The physical interlock provides backup to the mobility and positioning softwa constraints. Software may not work correctly while physical interlocks can and do fail on occasion. Neither alone provide adequate assurance.

[DP.3] Once the base reaches the work area and the stiff legs are deployed, the MA unfolcitself from its stowed configuration. All motions of the MA are designed to be manually operated should the need arise.
Rationale. In the course of periodic maintenance and servicing, it is expected that this feature wprove usefu..

[DP.4] The MA provides a number of motions, some redundant to reach the tiles. The firscalled Major-Z, raises the arm vertically. A second vertical motion is then used to lift the later sections of the MA. Atop this motion there is a 360 degree rotating motion.
Rationale. The two motions are used because a single telescoping device could not provide tl combination of stroke length, short unextended height, payload, and accuracy as neede d.

[DP.5] The MA is counterweighted and preloaded to keep the inspection and injectiotools steady. All MA motions have absolute encoding to give position at all times, even in the event of power cycling or computer failure.

Digital Camera

[DP.1] At least one digital camera is mounted in a position where it can provide the operator with information about the state of the area around the Tessellator robot. A second camera system is located on the MB and can transmit an image of the area in a 120 degree arc in front of the robot.
Rationale. The digital camera eliminates the need to make assumptions about whether the robtis within the visual field of the operator at all times or not. The camera providing an image from above provides the operator with a view of the obstacles in the work area all around the MB. The camera mounted on the MB itself looking forward provides the realtive position of the image with the direction the operator should move the joystick and thus can reduce erroneous joystick commands.

[DP.2] The DC images are sent directly to the operator interface.

Safety Fuse

[DP.1] The SF is located between the computer servo outputs and the motor amplifiers. monitors a large number of analog and digital sensor values, and functions as a smart fuse.

[DP.2] In the event that any parameter goes out of nominal operating ran, the SF removes power from the amplifiers and brakes, effectively locking the actuatuors.

[DP.3] If the SF goes into the HALT state, the operator can query the SF for the causof the shutdown and take corrective actions.

[DP.4] The SF can be reset by any command from the operator.

Proximity Sensing

[DP.1] Proximity sensing is used around the robot base and on sections cthe manipulator arm.

[DP.2] If the PSS senses anything too close to the robot, it sends thiinformation directly to the SF.
Rationale. The SF is alerted instead of the operator because the fuse can stop the robcfaster than the operator can detect the receipt of a message from the PSS and then release the deadman switch.

Aural and Visual Alerts

[DP.1] The visual alert system has colored flashing lights visible from all locations at a specified distance around the MB.

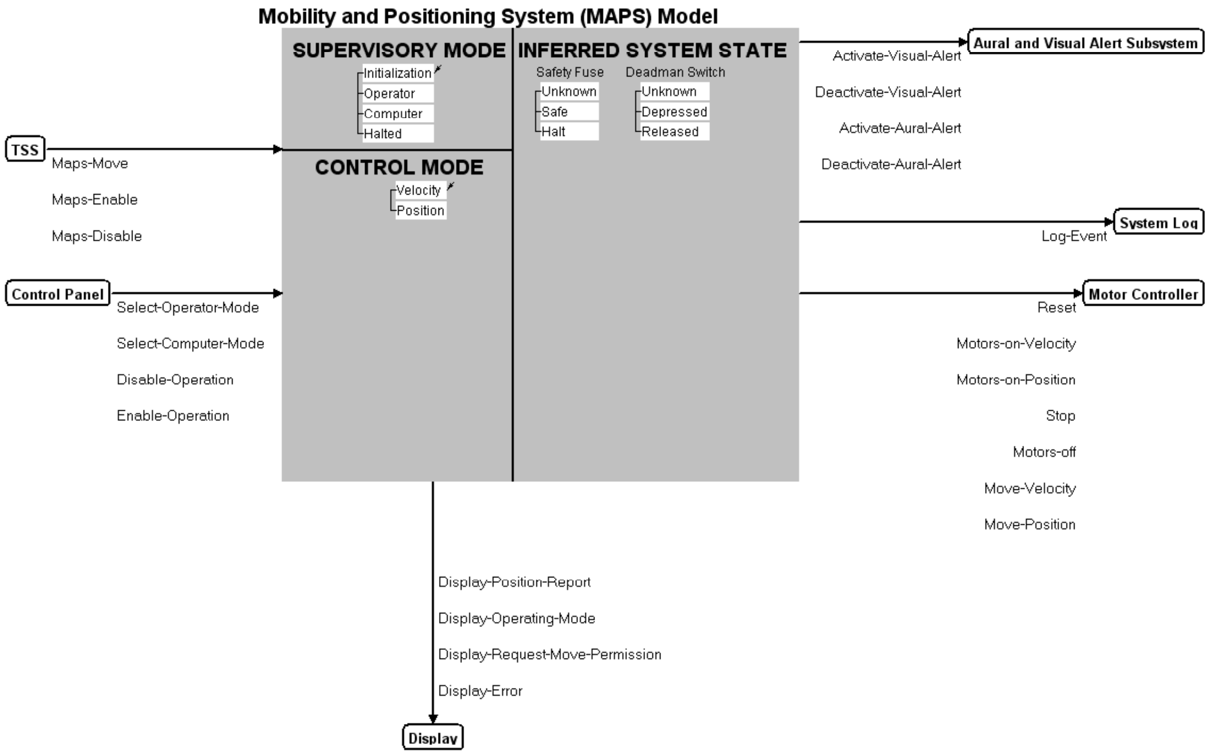
[DP.2] The aural alert system provides a sound that can be heard from anywhere in the vicinity of the MB.

Verification and Validation

This section includes requirements on and information about the validation of the design principles included in this level of the Intent Specification.

Level 3: Blackbox Behavior

System Blackbox Behavior - MAPS


Mobility and Positioning System (MAPS) Model
SUPERVISORY MODE
Initialization
Operator
Computer
Halted
INFERRED SYSTEM STATE
Safety Fuse
Unknown
Safe
Halt
Deadman Switch
Unknown
Depressed
Released
CONTROL MODE
Velocity
Position
TSS
Maps-Move
Maps-Enable
Maps-Disable
Control Panel
Select-Operator-Mode
Select-Computer-Mode
Disable-Operation
Enable-Operation
Aural and Visual Alert Subsystem
Activate-Visual-Alert
Deactivate-Visual-Alert
Activate-Aural-Alert
Deactivate-Aural-Alert
System Log
Log-Event
Motor Controller
Reset
Motors-on-Velocity
Motors-on-Position
Stop
Motors-off
Move-Velocity
Move-Position
Display-Position-Report
Display-Operating-Mode
Display-Request-Move-Permission
Display-Error
Display