

# Safety-Critical Requirements Specification and Analysis using SpecTRM

Grady Lee  
lee@safeware-eng.com

Jeffrey Howard  
howard@safeware-eng.com

Patrick Anderson  
patrick@safeware-eng.com

Safeware Engineering Corporation

## Abstract

SpecTRM (Specification Toolkit and Requirements Methodology) is designed to assist in the development of software-intensive safety-critical systems. Because most decisions that affect safety are made early in the product life cycle, SpecTRM focuses on system requirements and specification. SpecTRM uses a new approach to organizing system specifications, *intent specifications*, and the *SpecTRM-RL* formal modeling language. Intent specifications emphasize recording design rationale, the *why* of system specification, as well as what and how. A key feature of the intent specification is a blackbox model of the software behavior. This model is written in SpecTRM-RL (SpecTRM Requirements Language). SpecTRM-RL is founded on formalisms that support execution of the specification as well as automated safety analyses. However, SpecTRM-RL does not require training in mathematics to read. Domain experts can be taught to review SpecTRM-RL models in only a few minutes.

## Introduction

SpecTRM (Specification Toolkit and Requirements Methodology) assists in the development of software-intensive, safety-critical systems. SpecTRM focuses on system requirements and specification, where as much as 90% of the decision-making related to safety happens [2]. Too often, safety efforts focus on completed designs or implementations when changes are least effective and most costly. Even when safety efforts start early, the information generated by the hazard analysis may not be documented or provided to the system engineers at a time and in a form in which it can have the most effect on the developing system design.

SpecTRM was designed with several goals in mind:

- Integration of system safety into the system development process, particularly in the early stages of system development
- Traceability of hazard analysis into system and software requirements and design
- Reviewability of specifications by application experts and system safety engineers
- Completeness of requirements specifications
- Analyzability and assurance of critical properties.
- Recording of design rationale.

The requirements specification plays a pivotal role in the safety process. Specification review by domain experts is one of the least expensive and most effective techniques for enhancing safety. Almost all accidents related to software components in the past 20 years can be traced to flaws in the requirements specifications, such as unhandled cases.

Evolution of the system and component reuse also depend on the requirements and system specification. Changes and upgrades may violate safety-related design or environmental assumptions. When assumptions and design rationale are not recorded such that they can easily be found and used, changes to the system may lead to accidents. Reuse can be dangerous when the safety of the component is based on assumptions about the system in which it is used. Reuse puts the component into a new environment where those assumptions may be violated. Many accidents have resulted both from changes to operational systems and from design reuse between products. Any proposed changes must trigger a reanalysis of the safety of the system. Any reused components must be analyzed in the new environment. These potentially tedious and costly activities can be greatly simplified with a properly written specification.

# Process and Methodology

SpecTRM supports a safety-driven process for system specification and design. Because critical system properties such as safety must be designed into the system from the start, the methodology integrates safety analysis and human factors into the system design process from the beginning. This safety-driven human-centered approach was introduced by Leveson et al. [3]. This section gives a brief overview of the process and the resources that the SpecTRM methodology provides to enact this process: the intent specification and the SpecTRM-RL modeling language. Successive sections demonstrate how these three things (safety-driven process, intent specifications, and SpecTRM-RL) work together.

An outline of the methodology is shown in Figure 1. Because the process emphasizes safety and the consideration of the human-computer interface, activities related to those two focuses have been moved into their own columns. Note that the separation of process steps into three columns is merely to highlight the safety and human factors workflows. In practice, system safety and human factors activities must be closely integrated with system engineering tasks. Too often, safety and human factors are considered too late in the design of a system to have an impact. Similarly, the progress from the top of the column to the bottom is not just step-by-step. Any real process enactment involves iteration, skipping around, and refinement of the specification over the course of its development.

## Intent Specifications

The methodology is supported by an artifact called an intent specification [6]. Intent specifications are stratified into six levels, as shown in figure 2. In many common specification formats, a hierarchy of levels divides *what* to do (levels above) from *how* to do it (levels below). By contrast, intent specifications are not organized hierarchically along lines of refinement. Instead, each level is a completely different model of the same system. Each model presents a complete view of the system, but with a different intended audience, purpose, and notation or language.

**Level 0** provides a project management view and insight into the relationship between the plans and the project development.

**Level 1** of an intent specification is the customer view. It assists system engineers and customers to agree on what should be built and

whether that has been accomplished. Level 1 includes system goals, high-level requirements, design constraints, hazards, environmental assumptions, and system limitations.

**Level 2**, System Design Principles, is the system engineering level and allows engineers to reason about the system in terms of the physical principles and laws upon which the system design is based.

**Level 3**, the Blackbox Behavior level, enhances reasoning about the logical design of the system as a whole and interactions between components. This level acts as an unambiguous interface between systems engineering and component engineering. This level assists reasoning about component behavior using informal review, formal analysis, and simulation.

**Level 4** contains the design representation of the implementation of subsystem components. This level is written in a design notation appropriate to the component.

**Level 5** provides information necessary to reason about the physical implementation of a component. This may include code listings for short software modules, pointers to code repositories for larger software projects, or hardware assembly schematics.

**Level 6** provides a view of the operational system, including operator manuals, audit procedures, error reports, and change requests.

Each level is mapped to the levels above and below. Traceability links support reasoning across the levels and tracing from high-level requirements down to design and implementation details (and back again). Each level of the intent specification answers the question “*Why?*” for the level beneath. This intent information is the design rationale for the choices made in the specification, recorded directly in the specification. Each level also records the assumptions upon which the design and validation are based. Intent information is particularly important in safety-critical systems. When conditions change (e.g., the pacemakers designed for adults are to be used on children), a new safety analysis may be necessary. An intent specification makes finding the design decisions that depend on each assumption much less costly. The contents of an intent specification might look like those of figure 3.

## A Human-Centered, Safety-Driven Design Process

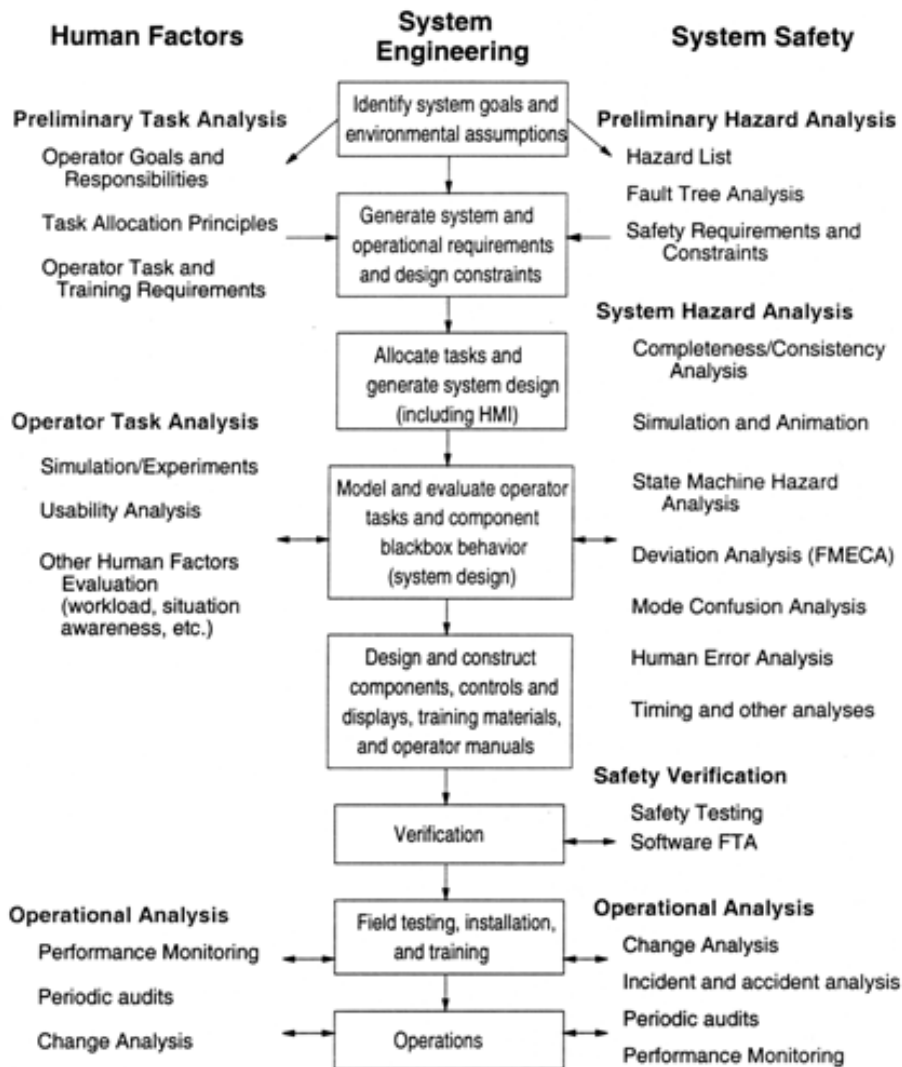


Figure 1: A Human-Centered and Safety-Driven Design Process.

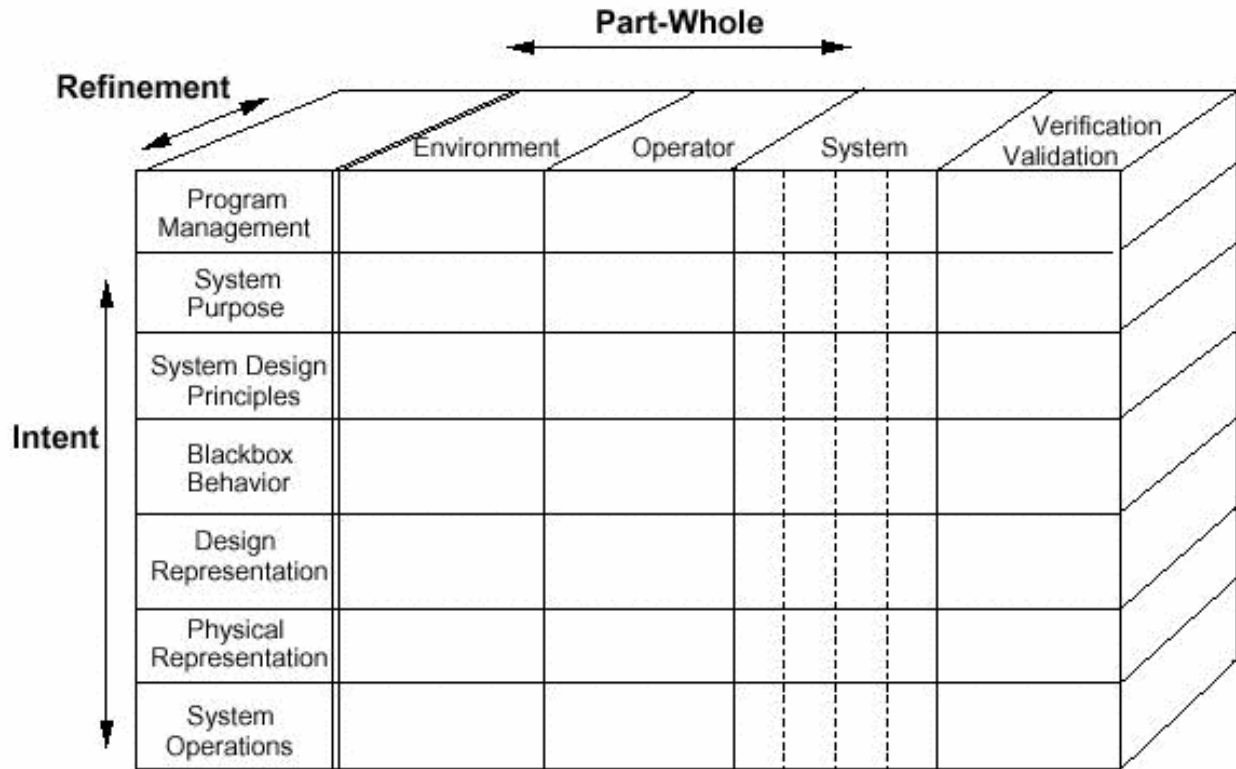


Figure 2: The Structure of an Intent Specification.

	Environment	Operator	System and components	V&V
<b>Level 0</b>	Project management plans, status information, safety plan, etc.			
<b>Level 1</b> System Purpose	Assumptions Constraints	Responsibilities Requirements I/F requirements	System goals, high-level requirements, design constraints, limitations	Preliminary Hazard Analysis Reviews
<b>Level 2</b> System Principles	External interfaces	Task analyses Task allocation Controls, displays	Logic principles, control laws, functional decomposition and allocation	Validation plan and results, System Hazard Analysis
<b>Level 3</b> Blackbox Models	Environment models	Operator Task models HCI models	Blackbox functional models Interface specifications	Analysis plans and results, Subsystem Hazard Analysis
<b>Level 4</b> Design Rep.		HCI design	Software and hardware design specs	Test plans and results
<b>Level 5</b> Physical Rep.		GUI design, physical controls design	Software code, hardware assembly instructions	Test plans and results
<b>Level 6</b> Operations	Audit procedures	Operator manuals Maintenance Training materials	Error reports, change requests, etc.	Performance monitoring and audits

Figure 3: Example Contents of an Intent Specification.

## SpecTRM-RL Modeling Language

One more feature is necessary to complete the methodology for creating a human-centered, safety-driven design process. SpecTRM specifications enhance analysis, simulation, and peer review. Intent specifications use a formal modeling language for describing the blackbox behavior of each system component (the transfer function). The modeling language has a formal (mathematical) basis to support formal and even automated analysis. In addition, the models can be executed allowing dynamic analysis of the specified system's behavior before any code is written. The design of the formal modeling language emphasizes readability so it can serve as a model and as the specification of the software requirements. Most project efforts do not have the resources for separate modeling and specification tasks, particularly in the face of changing requirements. SpecTRM-RL solves this problem as an easy to read specification language that can be executed and formally analyzed using automated tools.

Because SpecTRM-RL models can be executed, analyzed, and reviewed by domain experts, they admit the use of a wide variety of techniques to enhance the safety of the system being modeled. A SpecTRM-RL model in the context of an intent specification provides a seamless flow of rationale and reasoning from the highest level goals of the system, down through the SpecTRM-RL model all the way to the implementation and operator training materials. Using these tools, the human-centered safety-driven design process works to produce safer systems. These techniques for specifying the requirements for safety-critical systems have been validated on a number of projects, including air traffic control, autonomous robots, the automotive industry and aerospace.

The following sections walk through the process of creating an intent specification. Examples are used from selected parts of an intent specification for the movement and positioning software (MAPS) of an industrial robot [4]. The robot must move through a work area and deploy a manipulator arm to service heat shielding tiles on the Space Shuttle.

## The Management Level

The top level of an intent specification, Level 0: Program Management, presents a view of the project from a management perspective. The program management section should include the program management plan. Several standards cover management plans for software projects, such as IEEE Standard

1058. The exact form of the plan is less important than its incorporation into the intent specification.

The second part of the program management level is the System Safety Plan. The intent specification should have links from the plan to the parts of the specification that implement the plan. If the plan calls for the development of a hazard list, then the plan should link to where the hazard list has been compiled. A manager, customer, or regulatory body may use this section to ensure that the system safety plan is being implemented.

## System Goals and Preliminary Analyses

The system engineering process identifies the high level goals of the system. Goals are written into level 1 of the intent specification. Level 1 is a view of the system from the perspective of the customer; it includes contractual requirements and other high level purpose information. Derived requirements and design are placed later in the specification. Goals cannot act as requirements; they are not testable, nor are they specific enough to facilitate building a system. However, they are very important to understanding the system as they describe the mission to be achieved. Two high level goals from the industrial robot movement planning software specification are:

- G1:** MAPS shall control the movement of the robot around the work area and position it in the appropriate locations in the hangar so the tiles can be serviced ( $\rightarrow$  FR1).
- G2:** MAPS shall navigate according to commands from an operator-controlled, hand-held joystick or according to routes and destinations provided by an on-board computer (the Planner) ( $\rightarrow$  FR2, FR3, FR4).

**Assumption:** All robot base movement will be commanded from outside MAPS, either by the computer Planner or the operator.

Each goal is linked to the functional requirements that address that goal: FR1 in the case of G1. Because functional requirements are also at level 1 of the specification, the arrow points to the side. Arrows pointing up and down indicate up and down links to items at levels above and below the current level. In an electronic version of an intent specification, hyperlinks can be used to implement the links.

Additional information generated at level 1 of an intent specification includes the assumptions made

about the environment in which the system will operate. This information is important because changes to it should trigger a reevaluation of the system design including a safety analysis. An environmental assumption from MAPS is:

**EA1:** Upon request, the laser scanner will provide the current position of the robot in the form of global (world) location coordinates with an accuracy within 10 centimeters of the actual position (↓ 2.1.8, 2.4.1, 2.4.3.1, 2.4.5.4.2, 2.4.6.1, 2.4.2.6).

**Assumption:** The scanner will be used for position determination at the beginning of a move (the starting position) as well as determining the position of the robot after each segment of the route to allow for mid-course correction during planner-controlled movement.

**Assumption:** At least three bar code targets remain within the line of sight of the location system at all times.

## Preliminary Hazard Analysis

Preliminary hazard analysis begins with identification of system hazards. A number of methods have been proposed for discovering system hazards; they all involve focusing a group of people to apply domain specific knowledge to the task. At simplest, asking a number of *what-if* questions guided by the goals, environmental assumptions, and high-level requirements of the system may work. In practice, most systems will have few system level hazards. A preliminary hazard analysis of a national air traffic control system yielded only ten system level hazards. The hazards identified for MAPS are:

- H1** Violation of minimum separation between mobile base and objects (including orbiter and humans.)
- H2** Robot movement with stabilizers extended
- H3** Robot base becomes unstable
- H4** Manipulator arm hits something
- H5** Damage to robot caused by robot component operation or failure
- H6** Fire or Explosion
- H7** Contact of human with thermal tile waterproofing chemicals
- H8** Inadequate thermal protection

When an excess of hazards are identified, chances are that some of the proposed hazards are not actually hazards at all. They may instead be causes or

other contributing factors to genuine hazards. When the list of system hazards is complete, each hazard is given a hazard log entry. While the exact form of the entry should be tailored to the project, we draw our example from the MAPS specification:

**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 Causal 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 Planner;
- 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 (← SC1, FR2.4);
- Mobile base must stop when commanded(← SC2);
- Mobile base must not be commanded to an occupied position (← SC3);
- Operator must have information about objects in robot path and the ability to stop the mobile base quickly (← OP2, OP3, Disp1, Con4);
- Movement warnings must be provided (← FR6, SC9);
- Mobile base must not move if any safety-related subsystem is not operational (← SC8).

**Analyses Performed:**

**Actions Taken:**

**Status:**

**Verification:**

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

The hazard log links each hazard to safety constraints that eliminate or control the hazard. Once hazards are identified, developing these safety constraints is the next step. This is usually a straightforward process. The system must be constrained not to perform actions that lead to hazardous behavior. SC1 from MAPS reads:

**SC1:** The mobile base must move only when commanded by the operator or when commanded by the Planner and approved by the operator (→ H1)(← FR2.5).

**SC1.1:** MAPS must not enter Operator Mode unless the joystick is physically connected to the robot and the joystick is in the neutral position (↓ 2.4.6.1).

**SC1.2:** The robot must not move unless the deadman switch is depressed (↓2.4.2.1, 2.4.5.4.1, 2.4.6.3.3).

**SC1.3:** If the operator releases the deadman switch and then later depresses it again, all previous commands must be ignored and a new command must be issued before any robot movement occurs (↓ 2.4.2.5, 2.4.7.3, 2.4.5.4.1).

**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.

Constraints are linked down to design decisions in level 2 of the document. This reduces the cost of validating that the safety constraints have been implemented in the system by creating a direct path from the constraints to their realization. Note also that refinement occurs within the section. SC1.1 through SC1.3 are all refinements of SC1. In many specification formats, refinement occurs across levels of the specification, and intent information is lost altogether.

**Preliminary Task Analysis**

A Preliminary Task Analysis (PTA) is also performed very early on. The PTA involves human computer interaction experts dividing responsibilities

between the system and the operator. The resulting section of the intent specification provides information about the principles used to divide task responsibilities and allocations of each. The PTA also generates requirements and constraints on operator training manuals and procedures and can be used in the human-computer interaction and interface design process. An operator requirement derived from the Preliminary Task Analysis for MAPS is:

**OP1:** The operator shall supervise all robot base movement. (→ H1, H2, H3, H4, Con1).

**Assumption:** Jobs will be defined so that movement will occur approximately every half hour.

**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 than 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 tiles, time is affected primarily by time to service a tile and very little (in comparison) by the time to move the base.

**Generating the System Design**

Level 2 of the intent specification records the design of the system. The design of the system includes the basic physical laws and principles upon which the system depends. Level 2 describes how the requirements of level 1 are to be achieved, including any design features not related to level 1 requirements. The major sections of level 2 vary depending on project needs, but most systems include information such as the following:

**Interface Design** describes the messages that are sent between the system and other entities. The MAPS interface design describes input and output between MAPS, the user display, the user joystick, and a few other devices.

**Controls and Displays** describes the devices that the system will use to interact with the user.

**Operator Task Design Principles** lists the tasks the operator will be required to perform.

**System Design Principles** are the bulk of level 2.

A design principle from MAPS related to movement control reads:

**2.4.2.1** MAPS issues movement commands only if the safety fuse is in the SAFE state (↑ SC1.4.1), the manipulator arm is stowed (↑ EA8, SC6), the stiff legs are retracted (↑ SC5.1), the joystick is in the neutral position (↑ SC1.1), the operator has depressed the deadman switch (↑ SC1.2), and the safety fuse and motion alert system are both operational (↑ SC8.1). MAPS activates a visual alert 10 seconds before mobile base movement begins and an aural alert 5 seconds before movement begins (↑ FR6, SC9.1). Both are shutdown when in Computer Mode when the final destination is reached, when the deadman switch is released in either mode and in Operator Mode when the joystick is returned to a neutral position (↑ SC9.2).

**Verification and Validation** includes requirements on and information about the validation of the design principles. Verification and validation can include simulations, experiments, analyses, and other validation procedures including the system hazard analysis.

## Modeling and Evaluation

A model of the blackbox behavior of each component is used in analyses (both formal and informal) that evaluate the design. This is repeated in successive iterative rounds, feeding safety analysis results back into the black box behavior specification effort until the specification is complete.

The methodology uses formal models in a language called SpecTRM-RL (SpecTRM Requirements Language) to facilitate this evaluation process. Designers construct formal blackbox models of the required component behavior. The modeling language was designed with readability and reviewability by domain experts as a priority, so it does not require advanced mathematical training to use. Most reviewers can be taught how to read SpecTRM-RL in a few minutes.

SpecTRM-RL models include of the following elements:

**Outputs** describe data leaving the system.

**Modes** are clusters of operating behavior. Mode elements describe how the system switches between these major operating behaviors.

**State Values** represent state that the system infers about its environment from input data.

**Inputs** represent data coming into the system. All inputs have timing constraints to handle obsolescence. No data is good forever.

An example output element is shown in figure 4. This element shows a translation of system design principle 2.4.2.1 from above into a SpecTRM-RL model element. The output is sent if the triggering condition evaluates to true, which requires every row in the column to be true. If there were multiple columns in the table, any one of the columns evaluating as true would trigger a movement command (an “or” condition). In the tables, “T” stands for true, “F” for false, and “\*” for don’t care. So in the example, a movement command is sent if the conditions stated in the design principle labeled 2.4.2.1 (see above) is true. Note that the tabular model in Level 3 is linked to the English language description is Level 3, which in turn is linked to the high-level requirements and safety constraints.

A SpecTRM-RL model specifies the conditions under which each mode becomes active, the conditions under which outputs are produced and their content, and how each inferred state variable assumes a value. This information is sufficient to form an executable and formally analyzable model. An advantage of the executable specification over other methods (such as prototypes, special simulation languages, and mathematically intensive modeling languages) is that the specification is directly changed as the iterations of evaluation and changes to the system design proceed. At the end of the process, the final version of the model is still a highly readable specification. Implementation can proceed directly from the SpecTRM-RL. Rapid prototyping involves a costly reverse engineering step, and specialized simulation or mathematical modeling languages require extra effort to evolve in parallel with changes to the requirements.

SpecTRM uses several kinds of analysis to help evaluate alternatives as the model is developed. These are meant to be used in iterative rounds of specification and design, not just as a check at the end.

# MovementCommand

Output Command

**Destination:** Mobile Base

**Message:** MovementCommandMessage

**Timing Behavior:**

**Initiation Delay:** 2 seconds

**Completion Deadline:** this deadline is the largest straight line movement in the workspace divided by the maximum speed of the mobile base

**Output Capacity Assumptions:**

**Load:**

**Min time between outputs:** 2 milliseconds

**Max time between outputs:** none - the default behavior of the robot, which is not to move at all, is safe

**Feedback Information:**

**Variables:** Feedback from motion commands is read in through the laser positioning system

**Reversed By:** HaltMotion

**Description:** This output sends three values in a message: x, y, and theta. These three values are the next waypoint for motion. When a movement command is sent, the mobile base moves to obtain its new x and y position. The robot then rotates theta degrees.

**Comments:**

**References:** [1.2.4.2.1](#)

## TRIGGERING CONDITION

SafetyFuse in state Safe	T
ManipulatorArm in state Stowed	T
StiffLegs in state Retracted	T
JoystickX = 0	T
JoystickY = 0	T
DeadmanSwitch is Pressed	T
SafetyFuseState in state Operational	T
MotionAlertState in state Operational	T
Time Since MotionAlert Entered Visual > 10 seconds	T
Time Since MotionAlert Entered Aural > 5 seconds	T

## MESSAGE CONTENTS

Field	Value
X	WaypointX
Y	WaypointY
Theta	WaypointTheta

Figure 4: Example output specification.

## Completeness and Consistency Analysis

A goal for the design of SpecTRM-RL was to assist in writing complete specifications. Incompleteness (missing information or unusual conditions) is one of the most common causes of accidents involving software. Safeware [5] lists a set of completeness criteria for creating safe and complete requirements. SpecTRM-RL was designed to assist in enforcing these criteria through the specification language design. Most of the criteria that cannot be enforced through language syntax can be evaluated using automated tools.

Two of those criterion have to do with robustness and inconsistency of the specification. A specification is “robust” if there is a state transition or behavior specified for any input or set of inputs that may occur. This type of completeness check is easy to check automatically for SpecTRM-RL models, and instances where unusual but possible conditions are not handled can be identified.

Inconsistency is exactly the opposite problem. If more than one state transition is true at the same time, the software has no clear way to choose between the alternatives, and thus the specification is for a nondeterministic system. It is very difficult to assure that system properties such as safety hold for nondeterministic programs.

Robustness and consistency analysis can be used on a SpecTRM-RL model to find model elements that lack robustness and consistency [1]. Once identified, it is usually quite simple to add expressions to the table to eliminate such problems.

## Simulation and Animation

Because they are based on a state machine representation, it is possible to execute SpecTRM-RL specifications and evaluate the behavior of the system before it is designed and implemented. This dynamic analysis builds insight into how the system will behave when operational. An analyst could observe whether the MAPS specification always issues a “stop” command when the operator releases the deadman switch. Simulation is also a valuable technique on the human factors side of the process, allowing analysts to examine the system for behavior likely to lead to problems such as mode confusion. A simulation of the software using a SpecTRM-RL model can be integrated with simulation of other system components and the environment or even used in a hardware-in-the-loop simulation environment.

## State Machine Hazard Analysis

State machine hazard analysis (SMHA) is a backward search technique [7]. The analyst provides a description of a state known to be hazardous, and the analysis works backward to determine whether the unsafe state is reachable. Because the system hazards are traced to safety constraints and from there to elements in the model, the intent specification provides all the support necessary to easily generate these hazardous state conditions. SMHA could be used to answer the question, “Can the robot reach a state where the manipulator arm is extended but the stabilizer legs are not deployed.” If there is any path that makes this possible, the specification should obviously be changed to disallow it.

## Deviation Analysis

Software deviation analysis (SDA) is a technique for evaluating the robustness of a specification [8]. It examines the response of the software to deviations in software inputs, exploring software operation in imperfect environments. The analyst provides a list of deviations in inputs and identifies those outputs that are safety critical. The result of the analysis is a list of scenarios, which are combinations of input deviations and system states sufficient to cause a deviation in a safety-critical output, obviously an important flaw in the system design. One application of deviation analysis is to determine what might result if particular inputs from the environment are lost. An analyst might wish to answer the question, “How would MAPS be affected if the deadman switch became stuck while depressed by the operator during movement?” Starting with the deviation that the deadman switch is pressed when it should not be, deviation analysis can determine which safety critical outputs might be affected.

## Other Analyses

The preceding analyses are broadly applicable to a wide variety of software systems and tools can and have been written to perform them. Because the model underlying the SpecTRM-RL language is a formal one (a state machine), other types of automated analysis are possible to evaluate other properties appropriate to particular applications and projects. These analyses essentially can form the basis of a Subsystem Hazard Analysis. We are working on automating additional analysis techniques.

## Testing

Testing is also dependent on the form of the system being constructed. However, it is important that the test plan specifically include testing for safety. Because the intent specification maintains a path of traceability links starting at high level goals and hazards down to the implementation, the specification should readily support development of cases that test safety-critical behavior and provide traceability from the safety constraints to the test cases.

## Summary

The methodology described in this paper provides an approach to designing safety-critical systems where the design process is driven by safety considerations from the beginning. Because the most important system design decisions related to safety are made in the early stages of the system lifecycle, the methodology emphasizes requirements specification and system design.

The SpecTRM methodology is supported by intent specifications, a document that supports the full lifecycle of system and safety engineering and includes a formal and executable specification language. The informal parts of an intent specification provide traceability and the recording of design rationale. The formal specification language, SpecTRM-RL, was designed with readability and completeness of requirements specification as primary goals. Because SpecTRM-RL models are executable, they support simulation. Because they also are formal, they support a variety of formal and automatable analyses. We have developed tools to assist in the construction, execution, and analysis of intent specifications and SpecTRM-RL models and are working on improving these and creating additional analysis techniques and tools. We are also experimenting with additional analysis techniques and tools as well as ways to provide novel visualizations of the system requirements to assist in construction, expert review and safety analysis.

The SpecTRM methodology is practical and effective: it has been applied experimentally to a variety of systems, including collision avoidance, air traffic control, spacecraft control, and aircraft flight management. It has also been used commercially by SafeWare Engineering for real projects in the aerospace and automotive industries.

## References

- [1] Mats P.E. Heimdahl and Nancy G. Leveson. Completeness and consistency analysis of state-based requirements. *IEEE Transactions on Software Engineering*, May 1996.
- [2] William G. Johnson. *MORT Safety Assurance Systems*. Marcel Dekker, Inc., New York, 1980.
- [3] Nancy Leveson, Maxime de Villepin, Mirna Daouk, John Bellingham, Jayakanth Srinivasan, Natasha Neogi, Ed Bachelder, Nadine Pilon, and Geraldine Flynn. A safety and human-centered approach to developing new air traffic management tools. In *Proceedings of the ATM 2001*, December 2001.
- [4] Nancy G. Leveson. MAPS example intent specification.
- [5] Nancy G. Leveson. *Safeware: System Safety and Computers*. Addison Wesley, Reading, Massachusetts, 1995.
- [6] Nancy G. Leveson. Intent specifications: An approach to building human-centered specifications. *IEEE Transactions on Software Engineering*, SE-26(1), January 2000.
- [7] N. Neogi. *Hazard Elimination Using Backwards Reachability and Hybrid Modelling Techniques*. PhD thesis, Massachusetts Institute of Technology, 2001.
- [8] Jon Damon Reese. *Software Deviation Analysis*. PhD thesis, University of California Irvine, 1996.