AN INTENT SPECIFICATIONS MODEL FOR A ROBOTIC SOFTWARE CONTROL SYSTEM

Israel Navarro, Kristina Lundqvist, and Nancy Leveson

Massachusetts Institute of Technology, Cambridge, MA

Abstract

Intent specifications are a new way to structure specifications to support human problem solving, system and software development and evolution, traceability, and specification of design rationale. An intent specification provides a hierarchical abstraction based on intent ("why") in addition to the usual "what" and "how." For a given system being specified, an intent specification defines seven levels, each one of them supporting a different type of reasoning about the system. Each level is mapped to the appropriate parts of the intent levels above and below it, providing a means to trace design rationale and decisions from high-level system requirements and constraints down to code and vice versa (from code to specifications, requirements, and safety analyses). The third level of an intent specification contains a black-box model that uses an executable formal specification language, SpecTRM-RL, which provides special support for requirements review and analysis -- particularly for completeness and safety. SpecTRM-RL models can be mathematically analyzed and checked for various properties, including human-computer interaction properties such as mode confusion. They can also be executed as part of system simulations.

The approach is demonstrated using an industrial robot designed to service the heat resistant tiles on the Space Shuttle.

Introduction

Intent Specifications are a new way of structuring system and software specifications to support the development of large and complex real-time control systems [1]. Special attention is given to the support of system safety techniques throughout the entire development process.

The specifications are organized along two dimensions: intent abstraction and part-whole abstraction (see Figure 1). The intent dimension specifies seven hierarchical levels that each support a different type of reasoning about the system. The part-whole dimension is itself divided into refinement and decomposition, providing a way to structure the pertinent information within each level.

Lack of documentation and analysis of design decisions in any engineering project can lead to serious development delays and cost overruns, losses and disruptions during operations, and serious problems in upgrading and evolving the system design [2]. Specifying design rationale is particularly important to being able to change complex systems without introducing errors or inadvertently reversing decisions that should not be reversed. The fear of making such mistakes has on occasion led to leaving obsolete functions in the design (which, itself, has led to accidents). In either case, the safety of the system is compromised by the lack of documentation of necessary information.

The traditional specification models establish what-how relationships between the hierarchical levels. That is, they enumerate what functions are required and how they are accomplished. The intent or "why" relationship adds the ability to keep track of the rationale behind system design decisions and changes. Each intent level contains intent information ("why") about the level below. The levels are mapped to the appropriate parts of the intent levels above and below, providing

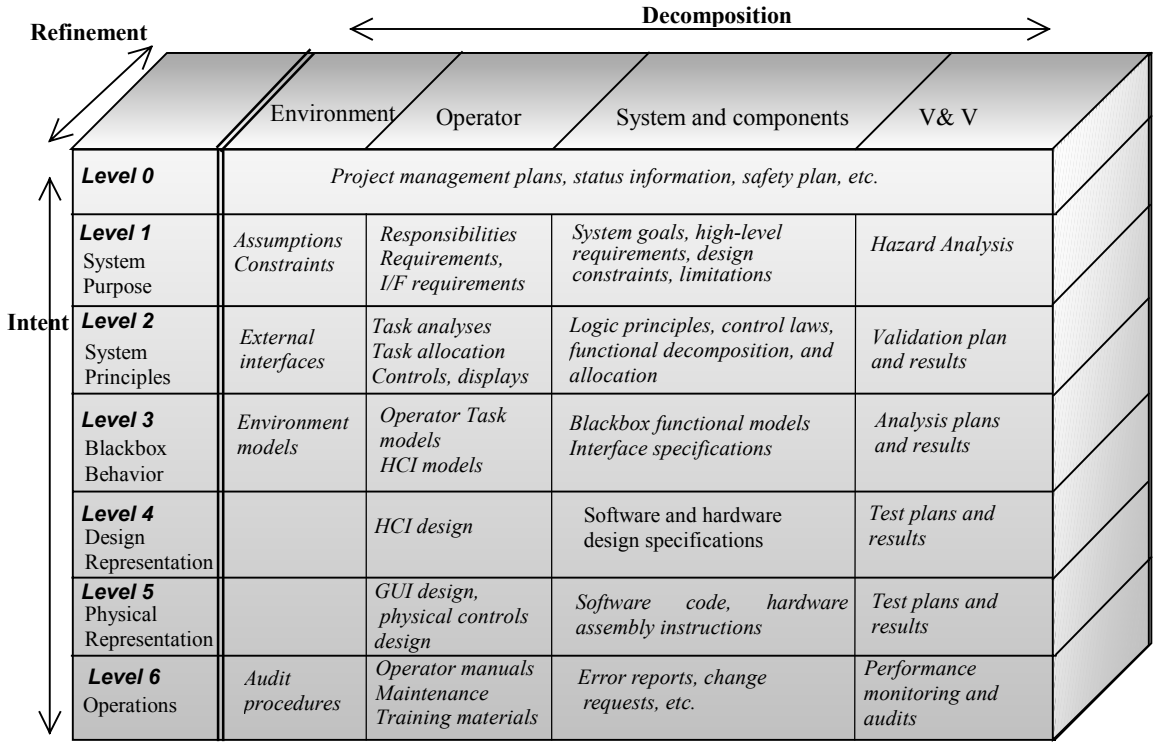

Figure 1: The structure of an intent specification for a complex software system

traceability of high-level system requirements and constraints all the way to code and vice versa.

Each intent level supports a different type of reasoning about the system. The Management level (level 0) provides a bridge from the contractual obligations and the management planning needs to the high-level engineering design plans. The System Purpose (level 1) assists system engineers in their reasoning about system-level goals, constraints, and limitations. It also documents basic hazard analysis and contains a hazard log to track the safety engineering activities. The system safety analysis is used to establish safety-related system and software requirements and design constraints.

The System Principles level (level 2) documents the system in terms of the physical principles and laws upon which the design is based. Complete functional structures and interfaces between components are defined at this point. This level also documents task models and other results of human factors analyses for systems containing human operators. Display and control information is defined as well.

The Blackbox level (level 3) allows the designers to study the logical design of the system using a formal modeling language called SpecTRM-RL [3]. The tools associated with this language produce executable and analyzable models that engineers can use to study the complex interactions between the different components of the system. SpecTRM-RL models can be built for the non-software components also, or they can be executed together with simulations and prototypes of hardware components.

The Design Representation level (level 4) presents a detailed implementation-dependent software design as well as any applicable hardware design specifications. The Physical Representation level (level 5) contains the actual software and hardware implementation of the system as well as

any necessary training and maintenance manuals. Finally, the System Operations level (level 6) includes information produced during the actual operation of the system, which can be used in operational audits, performance analyses, operational safety analyses, and change analysis.

Each level also contains documentation of the plans and results of the verification and validation appropriate for the design decisions contained in each level.

A goal behind intent specifications is to support a human-centered, safety-driven development process methodology. The specifications are human-centered not only in terms of supporting the design and integration of human-automation interaction into the system engineering process but also in basing the specification structure on what is known about how expert problem solvers and the most successful engineers approach system development tasks.

The specifications support safety-driven development by tightly integrating the system safety process and the information resulting from it into the system engineering process and decision-making environment. The goal is to support the design of safe systems rather than simply the attempt to verify safety after-the-fact. Safety-related design decisions are linked to hazard analyses and design implementations so that assurance of safety is also enhanced as well as any analyses required when changes are proposed.

Unlike most formal specification languages, the formal language used in Level 3, called SpecTRM-RL, has been designed with readability and reviewability and minimal training requirements (5 to 10 minutes) as the top priority, as well as a notation that is close to the way engineers think about systems.

Experiments have been conducted to determine the best way to present the information [4] and the experimental results have been validated by use of the language on several demonstration systems including a real Flight Management System [5] and an Air Traffic Control system upgrade project [6].

Various types of analysis techniques have been developed for SpecTRM-RL models including completeness and consistency analysis, a form of robustness analysis called Software Deviation Analysis, software hazard analysis, and mode confusion analysis [7] (for potential operator mode confusion caused or aggravated by the design of the automation).

The Tessellator Robot Example

In the next sections, an intent specification (levels 1 through 3 only) for a robotic system is described along with examples from the specification. The example system is not only safety-critical, but it includes jointly shared control of the system by computers and humans.

The Tessellator Robot

The underside of the Space Shuttle is covered by approximately 17,000 silica heat-resistant tiles that shield its aluminum skin from the 3000-degree Fahrenheit temperatures the orbiter encounters as it passes through Earth's atmosphere to land. These tiles have a glazed coating over soft and highly porous silica fibers. The tiles were designed to be extremely light and are 95% air by volume.

Unfortunately, this excellent design has a negative side effect: the tiles are extremely hydrophilic, being capable of absorbing enough water to create a substantial weight problem. To avoid it, the Space Shuttle tiles are waterproofed through the use of a specialized hydrophobic chemical, DMES, which must be injected into every tile. This process must be repeated after each mission because the waterproofing chemical burns off during the orbiter's re-entry into Earth's atmosphere.

This waterproofing task traditionally has been done manually. Dowling et al. [8] report on the creation of a robot, called Tessellator, to carry out these thermal protection tasks previously done by ground personnel. The main objective of their effort is to decrease the time needed to perform the waterproofing operations while increasing the safety of the overall ground operations tasks

(DMES is a toxic chemical, and workers have to wear protective gear on the job).

In addition, the robot is also designed to inspect each tile for possible damage. See Figure 2 for a picture of the robot developed by the CMU team.

Figure 2: Tessellator robot

The Tessellator has a computing environment consisting of four on-board computers and one off-board database. The on-board computers control the Tessellator's high-level processing tasks, base and manipulator motions; monitor robot health and status; and controls the robot's vision and injection systems. The software for the robot comprises a large part of the entire system. MAPS---the Mobility and Positioning Software---issues commands to the motor controller which controls movement of the mobile base of the robot. MAPS is in turn controlled either by a destination and route provided by the Planner (AI-based software) or by a human-operated joystick.

MAPS Intent Specification

For the purpose of demonstrating Intent Specifications, we will focus on MAPS only. This part of the robot software is of special interest because it is responsible for many of the safety-related functions. Intent specification examples are drawn only from levels 1, 2, and 3. Note that the original MAPS design has been modified to make it more interesting for study.

Level 1: System Purpose

Level 1 of an intent specification contains the goals, high-level functional requirements, constraints, and environmental assumptions corresponding to the system under study. The primary goal of MAPS is defined to be:

Goal 1: MAPS shall control the movement of the robot around the work area to position the robot base in the appropriate hangar locations so that the tiles can be serviced.

Examples of MAPS high-level functional requirements include:

MAPS-1.1: MAPS shall process control commands to several Tessellator subsystems (Motor Controller, Legs, Scanner) to ensure the robot base moves according to the requests issued by the Planner or the operator. [↓MAPS-2.2, ↓MAPS-2.2.2]

MAPS-1.1.1: MAPS shall be able to operate in Computer Mode (target position provided by the Planner). [↓MAPS-2.2.2, ↓MAPS-2.2.2.2, ↓MAPS-2.2.2.6, ↓MAPS-2.2.4]

Intent: The Planner is able to direct the robot efficiently so the robot meets all the required performance deadlines.

…

MAPS-1.1.1.3: While in Computer Mode, MAPS shall prompt the laser scanner automatically. [↓MAPS-2.2.5.7, ↓MAPS-2.2.2.5.7.1, ↓MAPS-2.2.2.5.7.2]

Intent: MAPS can use the position information to correct the robot trajectory and figure out when the final position has been reached.

…

MAPS-1.2: MAPS is responsible for sending messages to the system log about events and errors. [↓MAPS-2.3]

Intent: The recording of a variety of performance data will enable NASA system engineers fine-tune the software controlling the robot.

For traceability purposes, the high-level functional requirements are linked to the level 2 system design principles that satisfy them.

One of the major tasks in this part of the specification is to identify the boundaries of the system and enumerate assumptions the designers should use about the other components with which MAPS is to interact. Figure 3 presents an overview of the Tessellator system. Examples of environmental assumptions are:

Motor Controller

MC1: When commanded to do so, the motor controller will provide power to the motor, which will drive the robot wheels.

MC2: The robot may be driven in all three degrees of freedom (X,Y,θ), either singly or in any combination.

MC3: The robot may be driven in position (relative displacement) or velocity mode.

MC4: During power-up or following any error, the Motor Controller must be reset before sending it any orders.

MC5: The Motor Controller is able to stop the motion of the robot within 0.2 seconds of receiving a 'stop' command

Location System

LS1: The scanner must be initialized before first use with bar codes and scanner codes (the location coordinates of the barcode reflectors).

LS2: Upon request, the laser scanner will provide the current position of the robot in the form of global location coordinates with an accuracy of TBD.

LS2.1: The robot base must stop before a reading takes place.

Intent: The scanner triangulation calculations assume a static base and cannot correct their readings to take into account the robot base motion.

Additional assumptions were defined for other elements as well: Planner, stabilizers, safety circuit, manipulator arm, injection system, vision system, proximity-sensing system, digital camera, operator, displays and controls, and the work area. The assumption and requirements on the operator are especially important since they play a key role in the design of the operator interface, the modeling of the operator tasks and procedures as well as in the writing of training plans and programs.

Level 1 also contains the preliminary hazard analysis and hazard log. The first step in a system safety engineering process is to identify system hazards. For example, a MAPS-related hazard involves the loss of stability of the robot base:

***Hazard 3 (H3)**: Robot base becomes unstable*

***Subsystem**: MAPS, stabilizers*

***High-Level Causal Factors**: Stabilizers not deployed while arm extended; Stabilizers retracted while arm extended; Robot falls over while crossing cover*

***Level and Effect**:A2-4; Damage to Tessellator robot.*

***Safety Constraints**: [SC7] [SC8]*

From the list of hazards, the engineers must define safety-related system requirements and design constraints that ensure the robot never gets into one of the identified hazardous states. For example, the safety constraints that result from H3 are:

TESSELLATOR AND MAPS: SYSTEM OVERVIEW

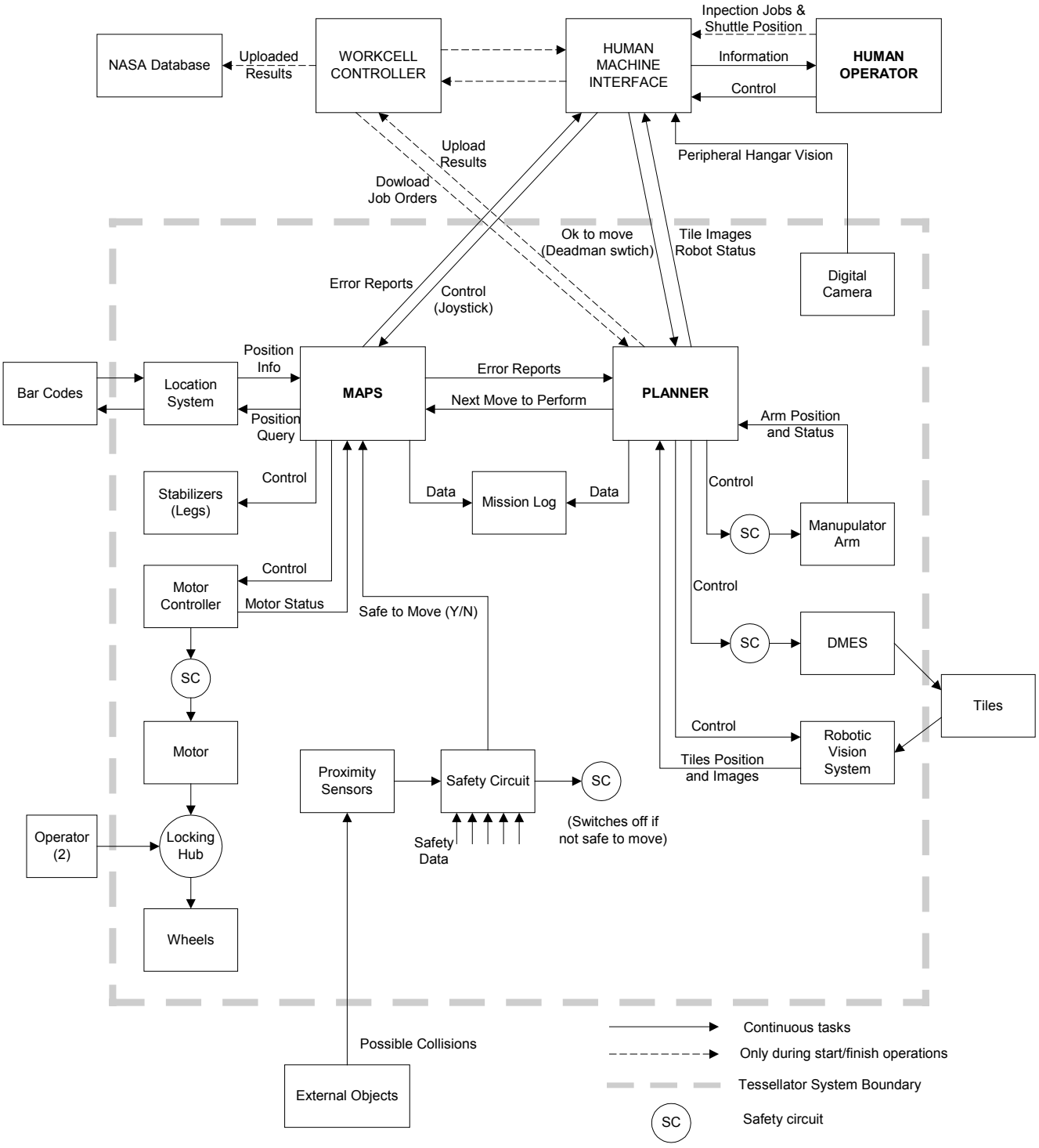

Figure 3: System overview of the Tessellator robotic system

[SC7]: Manipulator arm must move only when stabilizers are deployed. [↓MAPS-2.2.5.10.3]

[SC8]: Stabilizer legs must not be retracted until manipulator arm is fully stowed. [↓MAPS-2.2.5.10, ↓MAPS-2.2.6.9.2]

The safety constraints are then traced to those design features that ensure their satisfaction. This is of extreme importance because it allows designers to evaluate the safety of the original system design as well as to determine whether proposed design changes might affect system safety and thus need to be subjected to special analysis.

Level 2: System Design Principles

Level 2 of an intent specification documents the basic design principles upon which the system is built. It also provides the basis of the rationale behind the design decisions in the levels below. Finally, the functional structures established here describe how the requirements and constraints documented in Level 1 are satisfied.

One of the most important tasks in a complex system is the creation of an adequate interface design. For this, the project engineers must consider all the data dependencies among the software components of the system. Most likely, the interface design will be based on the environmental assumptions documented in level 1. Examples for MAPS include:

Planner/MAPS: The planner provides route of travel (in the form of a series of route segments) and destination to MAPS

Scanner/MAPS: The scanner sends current robot position information to MAPS when requested. MAPS will initialize the scanner with information about bar codes and scanner codes (the location coordinates of the barcode reflectors).

Level 2 also contains the basic principles behind the design of the controls and displays. The description of the human machine interface will include links to the assumptions and requirements on the operator tasks documented in level 1. In addition, a set of operator task design principles must be established. These describe the tasks for which the operator is responsible. A clear description of these procedures will enable the production of complete and accurate training requirements and manuals.

The core of the level 2 of an intent specification is the set of design principles that specify how the design will satisfy the requirements documented in level 1 while not violating any design constraints. The MAPS level 2 functional design principles show the functional decomposition upon which the software logic is structured. MAPS functionality is divided in four different operating modes (initialization, computer, operator, and safety). The functionality for each mode is designed to be independent of the others. This feature allows the designers to change the internal logic of one mode without worrying about the effect the changes will have on the other modes. The mode selection logic implements the mode transitions.

MAPS Design Principles

MAPS-2.1: System initialization will be performed immediately at startup and will be performed only once.

Intent: MAPS is responsible for initializing several robot subsystem prior to the beginning of normal operations

MAPS-2.2: MAPS shall generate all robot subsystems control commands from appropriate destination/movement inputs. [↑MAPS-1.1, ↑SC1]

MAPS-2.2.1: MAPS will not accept any motion commands until system initialization mode is complete.

Intent: The normal operation of the system relies on the correct performance of the subsystems initialized during startup.

MAPS-2.2.2: MAPS higher-level logic will provide a consistent way to determine under which mode the robot is to be controlled. [↑MAPS-1.1, ↑MAPS-1.1.1, ↑MAPS 1.1.2]

MAPS-2.2.2.1: MAPS will default to Operator mode of operation after initialization and after recovery from any type of temporary shutdown or movement inhibition (such as from the safety fuse). [↑MAPS-1.1.1.2, ↑MAPS-1.1.2, ↑MAPS-1.1.2.1, ↑SF2, ↑SC1]

Intent: The Operator Mode was chosen as default because only the human operator is capable of taking high-level decisions such as determining if the shuttle hangar area is free of obstacles or deciding whether the recovery procedure from an error has been 100% satisfactory.

...

MAPS-2.2.3: Under Safety Mode, MAPS will stop the robot, prevent further motions, and provide a set of recovery procedures. [↑SC2, ↑MC5]

MAPS-2.2.3.1: When informed by the safety fuse that motion is not legal; MAPS will cease all movement operations and inhibit any further movement operations until informed that movement is again allowed. [↑SC2, ↑SF1, ↑MC5]

MAPS-2.2.3.2: When the operator releases the dead-man switch, a rapid deceleration is required. This value shall default to TBD but should be changeable during operations. [↑Joystick4, ↑MC5, ↑SC2]

...

MAPS-2.2.5: When operating in Computer mode, MAPS will accept movement commands from the Planner and will issue all the necessary commands to move the robot. The route to the work area will be specified in the message from the planner as well as the final destination. [↑MAPS-1.1.1, ↑MAPS-1.1.1.1, ↑PL1.2, ↑SC1]

…

MAPS-2.2.5.7: MAPS shall query the laser scanner to obtain position information. [↑MAPS-1.1.1.3 , ↑LS2]

MAPS-2.2.5.7.1: Position determination will occur prior to the beginning of each route segment when operating in the Computer mode of operations. [↑MAPS-1.1.1.3, ↑LS2, ↑LS2.1]

MAPS-2.2.5.7.2: Position determination will occur following the completion of each route segment when operating in Computer Mode. [↑MAPS-1.1.1.3, ↑LS2, ↑LS2.1]

...

MAPS-2.3: All meaningful events, whether errors or successful moves, will be logged into a central log. [↑MAPS-1.2, ↑L1]

Each design principle is linked to the level 1 requirements, assumptions, or constraints on which that they depend. The design principles are also linked to the level 3 blackbox model (the downward links are not shown here).

Level 3: Black Box Behavior

Level 3 is designed to provide the system designers with a complete set of tools with which to validate the specified requirements before implementation begins. Only blackbox (externally visible) behavior is included, i.e., the input/output function to be computed by the component. In engineering terminology, this is sometimes called the transfer function across the component. Blackbox models assist in the requirements review process by eliminating implementation details that do not affect external behavior and thus are not relevant in the validation of the requirements. For this purpose, a model of the MAPS control software was produced using the formal specification language SpecTRM-RL.

SpecTRM-RL is built upon a traditional Mealy automaton although users of the model need not be familiar with the underlying mathematical model. The notation has been designed to be practical for specifying very large and complex systems and to include all the information needed to build such systems.

A SpecTRM-RL model has three components: (1) a specification of the supervisory interface to the component, (2) a specification of the control modes for the component, and (3) a model of the controlled process or plant including relevant operating modes, state variables, and interface variables (measured and manipulated process

variables as reflected by the inputs and outputs to the controller).

Once the model is built it can be validated using human review by domain experts, simulation and execution in a simulation environment, and analysis tools specially created for this type of model (as stated earlier, to evaluate completeness and consistency, hazardous behavior, robustness, and mode confusion).

Figure 5 shows the graphical part of the MAPS blackbox model. The graphical notation purposely mimics the typical engineering drawing of a control loop. The upper left quadrant (gray box in figure 5) displays the possible supervisory modes under which MAPS can operate. MAPS has two supervisory modes, depending on whether the operator or the Planner is responsible for controlling the robot movement.

The bottom left quadrant contains the operating modes for the controller itself. These are not internal states of the system but simply represent externally visible behavior about the controller's modes of operation.

The right half of the MAPS model represents inferred information about the states of the controlled systems. For MAPS, most of the state variables defined are related to the condition of the robot subsystems. The definition of how these state values change is essentially the specification of the control laws that MAPS must enforce.

Figure 4 shows an example of how legal state value changes are defined in SpecTRM-RL. The value of the Deadman Switch state depends on the input value received and the timing constraints defined on that input (shown elsewhere in the specification).

Notice the use of AND/OR tables in figure 4. An AND/OR table evaluates to true if any of its columns evaluates to true. A column is true if all of its rows that have a "T" are true and all of its rows with an "F" are false. Rows containing an asterisk represent "don't care" (or irrelevant) conditions. For the Deadman switch case, the "Pressed" table evaluates to true if the current corresponding input value is "Pressed". The Deadman switch "Unknown" table evaluates to true if the system is in startup (row1) or the Deadman switch input has become obsolete (row2). One of the analysis tools checks to make sure that the specification is deterministic, that is, only one column can be true at a time.

Note that all the system state variables in a SpecTRM-RL model are required to have an "Unknown" value. A very common error found in requirements specifications and often associated with accidents is assuming that the computer always has an accurate (up-to-date) model of the controlled system. If input processing and feedback is disrupted for some reason (including temporary halting of the computer itself), however, the assumed controlled-system state may inaccurately reflect the real state. In SpecTRM-RL models, each state variable defaults to the unknown value at startup and returns to the unknown value after interruptions in processing or expected (and necessary) inputs are not received.

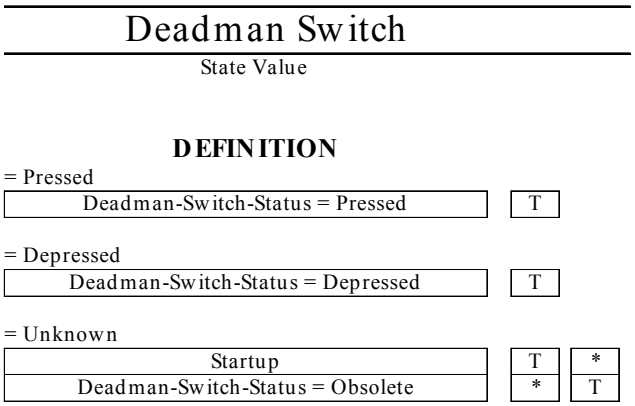

Figure 4: Example of state transitions definition

Returning to figure 5, note that the control inputs originating from the MAPS supervisor(s)

MODEL OF THE MAPS CONTROL SOFTWARE SYSTEM

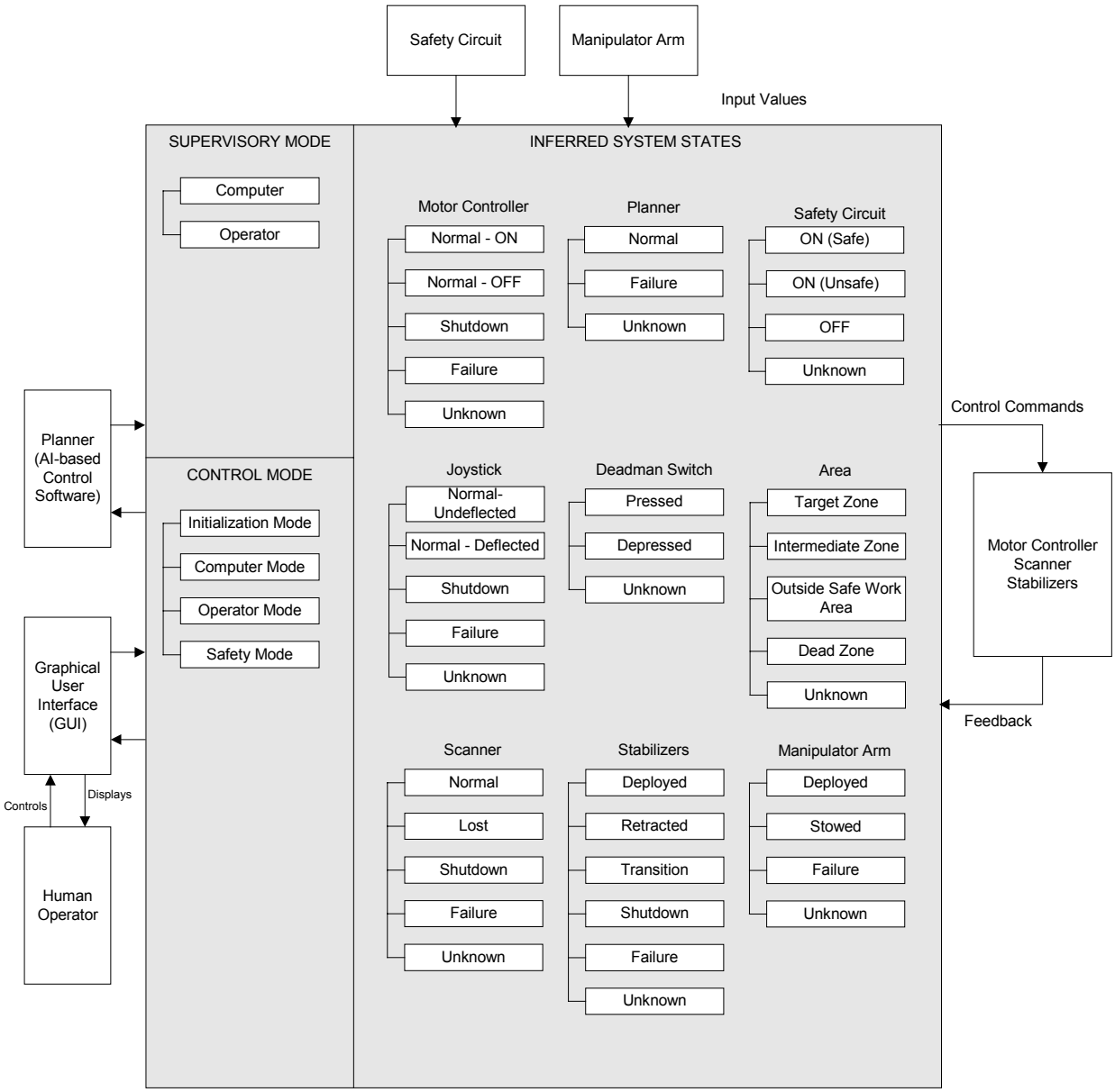

Figure 5: Graphical model of MAPS

Scanner Initialization Request

Output Command

Destination: Scanner
Acceptable Values: {Valid,Invalid}
Units: N.A.
Granularity: N.A.
Exception-Handling: N.A. (1-bit assumed)
Hazardous Values:
Timing Behavior:
Initiation Delay:
Completion Deadline:
Exception-Handling: Assumes value Invalid
Feedback Information:
Variables: Scanner-Acknowledgement, Scanner-Status
Values: {Valid,Invalid}, {Off,Ready,Lost,Failure} respectively
Relationship: Valid: Scanner-Acknowledgement{Valid} received within x seconds; Scanner-Status{Ready} or {Lost} received within y seconds (y>x assumed)
Min. time (latency):
Max. time:
Exception Handling: Scanner state set to value Failure
Reversed By:
Description: During startup (or after a scanner failure), MAPS must provide the proper commands to initialize the scanner. See also Scanner Initialization Data.
References: MAPS-2.1.3

DEFINITION

= Valid

MAPS Operating Mode in_state Initialization Mode	T	*
MAPS Operating Mode in_state Safety Mode	*	T
Scanner in_state Failure	*	T
Ready to initiate recovery	*	T

= Invalid

MAPS Operating Mode in_state Initialization Mode	F	F	F
MAPS Operating Mode in_state Safety Mode	F	*	*
Scanner in_state Failure	*	F	*
Ready to initiate recovery	*	*	F

Figure 6: Example of an output command

(i.e., the human operator and the Planner) are shown to the left of the MAPS model. Inputs coming from simple sensors (like the safety circuit) are shown above the model. Finally, the output commands to the devices being controlled (like the Motor Controller or the scanner) are shown to the right of the MAPS model.

An example output command specification is shown in figure 6. Besides the AND/OR tables already discussed, the output command specifications have a number of fields containing information critical to the completeness of the specification. We have identified about 60 completeness criteria for requirements specifications that are particularly related to safety. Industrial projects have been using these successfully in a checklist format, but such checklists have drawbacks. To assist in evaluating completeness and in reviewing requirements, the SpecTRM-RL modeling language either includes the information necessary to satisfy most of these

completeness criteria or simple tools can check them automatically.

For example, accidents have resulted when software did not include checks on the feedback information provided about the effect of previous output commands. Such feedback information (an example is shown in Figure 6) makes engineers consider carefully not only what their expectations are regarding the nature of the system interactions, but also forces them to provide a backup plan for those occasions when the expectations are not met. Overall, the blackbox SpecTRM-RL language encourages models to be built that help engineers focus their efforts on those issues fundamental to the safety of the system.

Conclusions and Future Work

The goal of intent specifications is to support the systems engineering techniques fundamental to the successful development of large and complex software-controlled systems. The formal specification language SpecTRM-RL, used for blackbox behavioral modeling on Level 3, provides engineers with a tool to review and validate requirements specifications.

References

[1] Leveson, N.G., "Intent Specifications: An Approach to Building Human-Centered Specifications," *IEEE Transactions on Software Engineering*, Vol. 26, No. 1, pp. 15-35, January 2000.

[2] Leveson, N.G., *Safeware: System Safety and Computers*, Addison-Wesley Publishing Company, 1995.

[3] Leveson N.G., "Completeness in Formal Specification Language Design for Process-Control Systems," ACM Formal Methods in Software Practice, Portland, August 2000.

[4] Zimmerman, Marc, *Investigating the Readability of Formal Specification Languages*, MIT Aeronautics and Astronautics S.M. Thesis, May 2001

[5] Zimmerman M., M. Rodriguez, B. Ingram, M. Katahira, M. de Villepin, N.G. Leveson. "Making Formal Methods Practical," Digital Aviation Systems Conference, October 2000.

[6] Leveson, N.G. et al., "Demonstration of a Safety Analysis on a Complex System", Software Engineering Laboratory Workshop, NASA Goddard, December 1997.

[7] Rodriguez, M., M. Zimmerman, M. Katahira, Maxime de Villepin, B. Ingram, and N.G. Leveson, "Identifying Mode Confusion Potential in Software Design", Digital Aviation Systems Conference (DASC), October 2000.

[8] Dowling, K.R., R Bennett, M. Blackwell, T. Graham, S. Gatrall, R. O'Toole, and H. Schempf, "A Mobile Robot System for Ground Servicing Operations on the Space Shuttle", *Cooperative intelligent robotics in space III*; Proceedings of the Meeting, Boston, MA, Nov. 16-18, 1992 (A93-29101 10-54). Bellingham, WA, Society of Photo-Optical Instrumentation Engineers, 1992, p. 298-309.