10 Integrating Safety into System Engineering

Previous chapters have provided the individual pieces of the solution to engineering a safer world. This chapter demonstrates how to put these pieces together to integrate safety into a system engineering process. No one process is being proposed: Safety must be part of any system engineering process.

The glue that integrates the activities of engineering and operating complex systems is specifications and the safety information system. Communication is critical in handling any emergent property in a complex system. Our systems today are designed and built by hundreds and often thousands of engineers and then operated by thousands and even tens of thousands more people. Enforcing safety constraints on system behavior requires that the information needed for decision making is available to the right people at the right time, whether during system development, operations, maintenance, or reengineering.

This chapter starts with a discussion of the role of specifications and how systems theory can be used as the foundation for the specification of complex systems. Then an example of how to put the components together in system design and development is presented. Chapters 11 and 12 cover how to maximize learning from accidents and incidents and how to enforce safety constraints during operations. The design of safety information systems is discussed in chapter 13.

10.1 The Role of Specifications and the Safety Information System

While engineers may have been able to get away with minimal specifications during development of the simpler electromechanical systems of the past, specifications are critical to the successful engineering of systems of the size and complexity we are attempting to build today. Specifications are no longer simply a means of archiving information; they need to play an active role in the system engineering process. They are a critical tool in stretching our intellectual capabilities to deal with increasing complexity.

Our specifications must reflect and support the system safety engineering process and the safe operation, evolution and change of the system over time. Specifications should support the use of notations and techniques for reasoning about hazards and safety, designing the system to eliminate or control hazards, and validating—at each step, starting from the very beginning of system development—that the evolving system has the desired safety level. Later, specifications must support operations and change over time.

Specification languages can help (or hinder) human performance of the various problem-solving activities involved in system requirements analysis, hazard analysis, design, review, verification and validation, debugging, operational use, and maintenance and evolution (sustainment). They do this by including notations and tools that enhance our ability to: (1) reason about particular properties, (2) construct the system and the software in it to achieve them, and (3) validate—at each step, starting from the very beginning of system development—that the evolving system has the desired qualities. In addition, systems and particularly the software components are continually changing and evolving; they must be designed to be changeable and the specifications must support evolution without compromising the confidence in the properties that were initially verified.

Documenting and tracking hazards and their resolution are basic requirements for any effective safety program. But simply having the safety engineer track them and maintain a hazard log is not enough—information must be derived from the hazards to inform the system engineering process and that information needs to be specified and recorded in a way that has an impact on the decisions made during system design and operations. To have such an impact, the safety-related information required by the engineers needs to be *integrated into* the environment in which safety-related engineering decisions are made. Engineers are unlikely to be able to read through volumes of hazard analysis information and relate it easily to the specific component upon which they are working. The information the system safety engineer has generated must be presented to the system designers, implementers, maintainers, and operators in such a way that they can easily find what they need to make safer decisions.

Safety information is not only important during system design; it also needs to be presented in a form that people can learn from, apply to their daily jobs, and use throughout the life cycle of projects. Too often, preventable accidents have occurred due to changes that were made after the initial design period. Accidents are frequently the result of safe designs becoming unsafe over time when changes in the system itself or in its environment violate the basic assumptions of the original hazard analysis. Clearly, these assumptions must be recorded and easily retrievable when changes occur. Good documentation is the most important in complex systems where nobody is able to keep all the information necessary to make safe decisions in their head.

What types of specifications are needed to support humans in system safety engineering and operations? Design decisions at each stage must be mapped into the goals and constraints they are derived to satisfy, with earlier decisions mapped or traced to later stages of the process. The result should be a seamless and gapless record of the progression from high-level requirements down to component requirements and designs or operational procedures. The rationale behind the design decisions needs to be recorded in a way that is easily retrievable by those reviewing or changing the system design. The specifications must also support the various types of formal and informal analysis used to decide between alternative designs and to verify the results of the design process. Finally, specifications must assist in the coordinated design of the component functions and the interfaces between them.

The notations used in specification languages must be easily readable and learnable. Usability is enhanced by using notations and models that are close to the mental models created by the users of the specification and the standard notations in their fields of expertise.

The structure of the specification is also important for usability. The structure will enhance or limit the ability to retrieve needed information at the appropriate times.

Finally, specifications should not limit the problem-solving strategies of the users of the specification. Not only do different people prefer different strategies for solving problems, but the most effective problem solvers have been found to change strategies frequently [167, 58]. Experts switch problem-solving strategy when they run into difficulties following a particular strategy and as new information is obtained that changes the objectives or subgoals or the mental workload needed to use a particular strategy. Tools often limit the strategies that can be used, usually implementing the favorite strategy of the tool designer, and therefore limiting the problem solving strategies supported by the specification.

One way to implement these principles is to use *intent specifications* [120].

10.2 Intent Specifications

Intent specifications are based on systems theory, system engineering principles, and psychological research on human problem solving and how to enhance it. The goal is to assist humans in dealing with complexity. While commercial tools exist that implement intent specifications directly, any specification languages and tools can be used that allow implementing the properties of an intent specification.

An intent specification differs from a standard specification primarily in its structure, not its content: no extra information is involved that is not commonly found in detailed specifications—the information is simply organized in a way that has been found to assist in its location and use. Most complex systems have voluminous documentation, much of it redundant or inconsistent, and it degrades quickly as changes are made over time. Sometimes important information is missing, particularly information about *why* something was done the way it was—the intent or design rationale. Trying to determine whether a change might have a negative impact on safety, if possible at all, is usually enormously expensive and often involves regenerating analyses and work that was already done but either not recorded or not easily located when needed. Intent specifications were designed to help with these problems: Design rationale, safety analysis results, and the assumptions upon which the system design and validation are based are integrated directly into the system specification and its structure, rather than stored in separate documents, so the information is at hand when needed for decision making.

The structure of an intent specification is based on the fundamental concept of hierarchy in systems theory (see chapter 3) where complex systems are modeled in terms of a hierarchy of levels of organization, each level imposing constraints on the degree of freedom of the components at the lower level. Different description languages may be appropriate at the different levels. Figure 10.1 shows the seven levels of an intent specification.

Intent specifications are organized along three dimensions: intent abstraction, part-whole abstraction, and refinement. These dimensions constitute the problem space in which the human navigates. Part-whole abstraction (along the horizontal dimension) and refinement (within each level) allow users to change their focus of attention to more or less detailed views within each level or model. The vertical dimension specifies the level of intent at which the problem is being considered.

Each intent level contains information about the characteristics of the environment, human operators or users, the physical and functional system components, and requirements for and results of verification and validation activities for that level. The safety information is embedded in each level, instead of being maintained in a separate safety log, but linked together so that it can easily be located and reviewed.

The vertical intent dimension has seven levels. Each level represents a different model of the system from a different perspective and supports a different type of reasoning about it. Refinement and decomposition occurs within each level of the specification, rather than between levels. Each level provides information not just about *what* and *how*, but *why*, that is, the design rationale and reasons behind the design decisions, including safety considerations.

Figure 10.2 shows an example of the information that might be contained in each level of the intent specification.

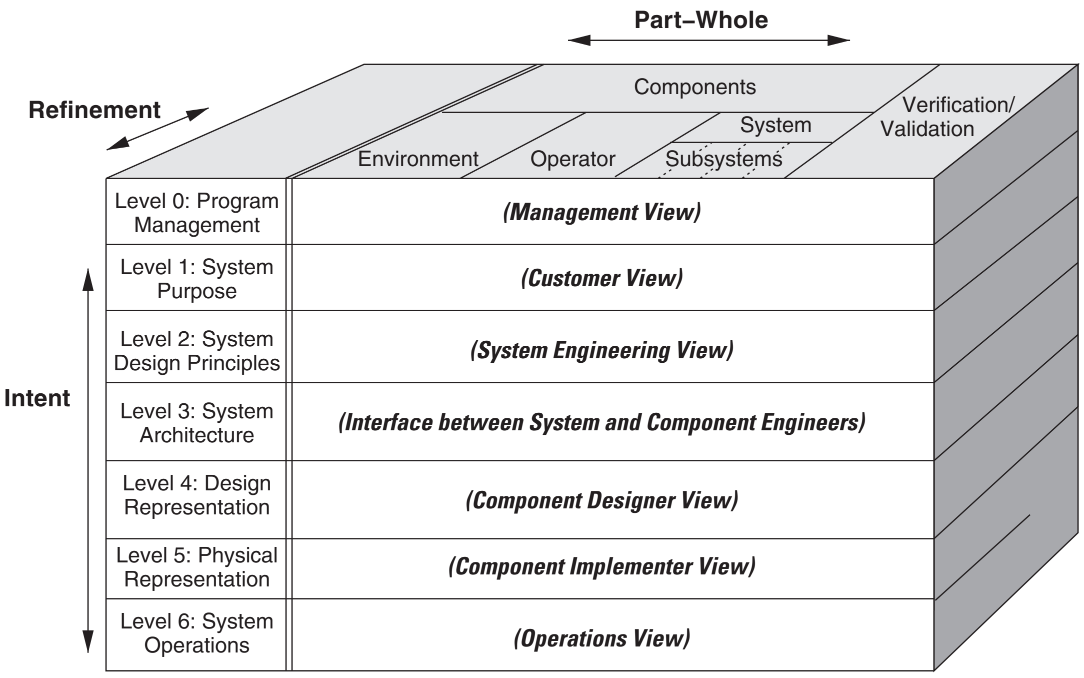

Part–Whole
Refinement
Components
Environment
Operator
System
Subsystems
Verification/ Validation
Intent
Level 0: Program Management
(Management View)
Level 1: System Purpose
(Customer View)
Level 2: System Design Principles
(System Engineering View)
Level 3: System Architecture
(Interface between System and Component Engineers)
Level 4: Design Representation
(Component Designer View)
Level 5: Physical Representation
(Component Implementer View)
Level 6: System Operations
(Operations View)

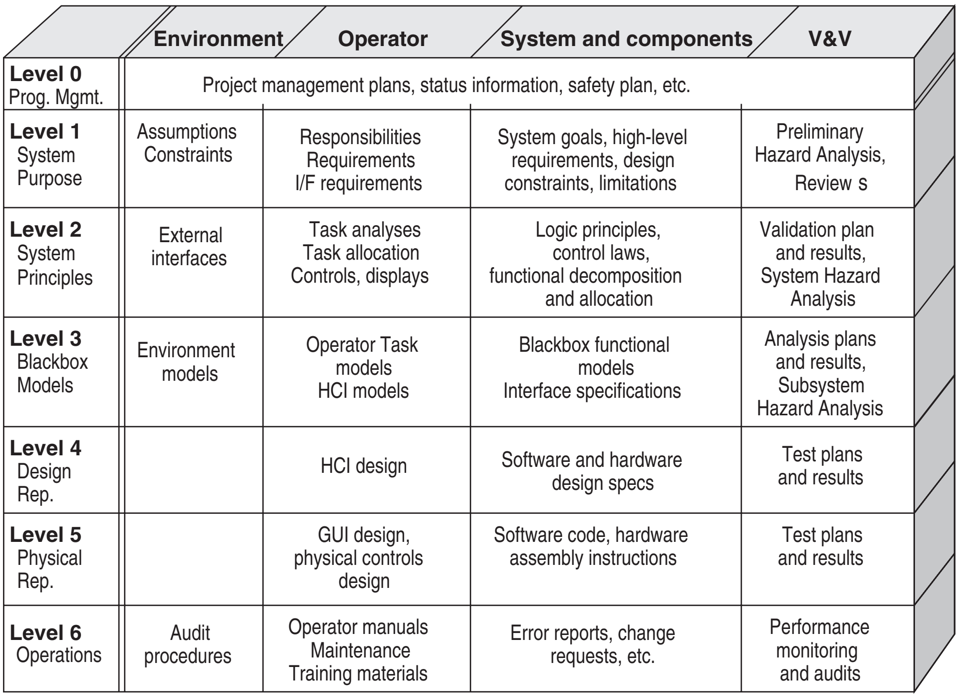
Figure 10.1
The structure of an intent specification.

The top level (level 0) provides a project management view and insight into the relationship between the plans and the project development status through links to the other parts of the intent specification. This level might contain the project management plans, the safety plan, status information, and so on.

Level 1 is the customer view and assists system engineers and customers in agreeing on what should be built and, later, whether that has been accomplished. It includes goals, high-level requirements and constraints (both physical and operator), environmental assumptions, definitions of accidents, hazard information, and system limitations.

Level 2 is the system engineering view and helps system engineers record and reason about the system in terms of the physical principles and system-level design principles upon which the system design is based.

Level 3 specifies the system architecture and serves as an unambiguous interface between system engineers and component engineers or contractors. At level 3, the system functions defined at level 2 are decomposed, allocated to components, and specified rigorously and completely. Black-box behavioral component models may be used to specify and reason about the logical design of the system as a whole and



	Environment	Operator	System and components	V&V
Level 0 Prog. Mgmt.	Project management plans, status information, safety plan, etc.			
Level 1 System Purpose	Assumptions Constraints	Responsibilities Requirements I/F requirements	System goals, high-level requirements, design constraints, limitations	Preliminary Hazard Analysis, Review s
Level 2 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
Level 3 Blackbox Models	Environment models	Operator Task models HCI models	Blackbox functional models Interface specifications	Analysis plans and results, Subsystem Hazard Analysis
Level 4 Design Rep.		HCI design	Software and hardware design specs	Test plans and results
Level 5 Physical Rep.		GUI design, physical controls design	Software code, hardware assembly instructions	Test plans and results
Level 6 Operations	Audit procedures	Operator manuals Maintenance Training materials	Error reports, change requests, etc.	Performance monitoring and audits

Figure 10.2
An example of the information in an intent specification.

the interactions among individual system components without being distracted by implementation details.

If the language used at level 3 is formal (rigorously defined), then it can play an important role in system validation. For example, the models can be executed in system simulation environments to identify system requirements and design errors early in development. They can also be used to automate the generation of system and component test data, various types of mathematical analyses, and so forth. It is important, however, that the black-box (that is, transfer function) models be easily reviewed by domain experts—most of the safety-related errors in specifications will be found by expert review, not by automated tools or formal proofs.

A readable but formal and executable black-box requirements specification language was developed by the author and her students while helping the FAA specify the TCAS (Traffic Alert and Collision Avoidance System) requirements [123]. Reviewers can learn to read the specifications with a few minutes of instruction about the notation. Improvements have been made over the years, and it is being used successfully on real systems. This language provides an existence case that a readable and easily learnable but formal specification language is possible. Other languages with the same properties, of course, can also be used effectively.

The next two levels, Design Representation and Physical Representation, provide the information necessary to reason about individual component design and implementation issues. Some parts of level 4 may not be needed if at least portions of the physical design can be generated automatically from the models at level 3.

The final level, Operations, provides a view of the operational system and acts as the interface between development and operations. It assists in designing and performing system safety activities during system operations. It may contain required or suggested operational audit procedures, user manuals, training materials, maintenance requirements, error reports and change requests, historical usage information, and so on.

Each level of an intent specification supports a different type of reasoning about the system, with the highest level assisting systems engineers in their reasoning about system-level goals, constraints, priorities, and tradeoffs. The second level, System Design Principles, allows engineers to reason about the system in terms of the physical principles and laws upon which the design is based. The Architecture level enhances reasoning about the logical design of the system as a whole, the interactions between the components, and the functions computed by the components without being distracted by implementation issues. The lowest two levels provide the information necessary to reason about individual component design and implementation issues. The mappings between levels provide the relational information that allows reasoning across hierarchical levels and traceability of requirements to design.

Hyperlinks are used to provide the relational information that allows reasoning within and across levels, including the tracing from high-level requirements down to implementation and vice versa. Examples can be found in the rest of this chapter.

The structure of an intent specification does not imply that the development must proceed from the top levels down to the bottom levels in that order, only that at the end of the development process, all levels are complete. Almost all development involves work at all of the levels at the same time.

When the system changes, the environment in which the system operates changes, or components are reused in a different system, a new or updated safety analysis is required. Intent specifications can make that process feasible and practical.

Examples of intent specifications are available [121, 151] as are commercial tools to support them. But most of the principles can be implemented without special tools beyond a text editor and hyperlinking facilities. The rest of this chapter assumes only these very limited facilities are available.

10.3 An Integrated System and Safety Engineering Process

There is no agreed upon best system engineering process and probably cannot be one—the process needs to match the specific problem and environment in which it is being used. What is described in this section is how to integrate safety engineering into *any* reasonable system engineering process.

The system engineering process provides a logical structure for problem solving. Briefly, first a need or problem is specified in terms of objectives that the system must satisfy and criteria that can be used to rank alternative designs. Then a process of system synthesis takes place that usually involves considering alternative designs. Each of the alternatives is analyzed and evaluated in terms of the stated objectives and design criteria, and one alternative is selected. In practice, the process is highly iterative: The results from later stages are fed back to early stages to modify objectives, criteria, design decisions, and so on.

Design alternatives are generated through a process of system architecture development and analysis. The system engineers first develop requirements and design constraints for the system as a whole and then break the system into subsystems and design the subsystem interfaces and the subsystem interface topology. System functions and constraints are refined and allocated to the individual subsystems. The emerging design is analyzed with respect to desired system performance characteristics and constraints, and the process is iterated until an acceptable system design results.

The difference in safety-guided design is that hazard analysis is used throughout the process to generate the safety constraints that are factored into the design decisions as they are made. The preliminary design at the end of this process must be described in sufficient detail that subsystem implementation can proceed independently. The subsystem requirements and design processes are subsets of the larger system engineering process.

This general system engineering process has some particularly important aspects. One of these is the focus on interfaces. System engineering views each system as an integrated whole even though it is composed of diverse, specialized components, which may be physical, logical (software), or human. The objective is to design subsystems that when integrated into the whole provide the most effective system possible to achieve the overall objectives. The most challenging problems in building complex systems today arise in the interfaces between components. One example is the new highly automated aircraft where most incidents and accidents have been blamed on human error, but more properly reflect difficulties in the collateral design of the aircraft, the avionics systems, the cockpit displays and controls, and the demands placed on the pilots.

A second critical factor is the integration of humans and nonhuman system components. As with safety, a separate group traditionally does human factors design and analysis. Building safety-critical systems requires integrating both system safety and human factors into the basic system engineering process, which in turn has important implications for engineering education. Unfortunately, neither safety nor human factors plays an important role in most engineering education today.

During program and project planning, a system safety plan, standards, and project development safety control structure need to be designed including policies, procedures, the safety management and control structure, and communication channels. More about safety management plans can be found in chapters 12 and 13.

Figure 10.3 shows the types of activities that need to be performed in such an integrated process and the system safety and human factors inputs and products. Standard validation and verification activities are not shown, since they should be included throughout the entire process.

The rest of this chapter provides an example using TCAS II. Other examples are interspersed where TCAS is not appropriate or does not provide an interesting enough example.

10.3.1 Establishing the Goals for the System

The first step in any system engineering process is to identify the goals of the effort. Without agreeing on where you are going, it is not possible to determine how to get there or when you have arrived.

TCAS II is a box required on most commercial and some general aviation aircraft that assists in avoiding midair collisions. The goals for TCAS II are to:

G1: *Provide affordable and compatible collision avoidance system options for a broad spectrum of National Airspace System users.*

G2: *Detect potential midair collisions with other aircraft in all meteorological conditions; throughout navigable airspace, including airspace not covered by ATC primary or secondary radar systems; and in the absence of ground equipment.*

TCAS was intended to be an independent backup to the normal Air Traffic Control (ATC) system and the pilot's "see and avoid" responsibilities. It interrogates air traffic control transponders on aircraft in its vicinity and listens for the transponder replies. By analyzing these replies with respect to slant range and relative altitude, TCAS determines which aircraft represent potential collision threats and provides appropriate display indications, called advisories, to the flight crew to assure proper

Agree on system goals

Identify constraints on how goals can be achieved
- Define accidents (unacceptable losses)
- Identify hazards
- Formulate system-level safety and non-safety constraints

Select a system architecture
- Architectural trade analysis
- Preliminary hazard analysis

Identify environmental assumptions

Create a concept of operations
Perform a preliminary operator task analysis

Refine goals into testable and achievable system-level functional requirements

Refine safety constraints and functional requirements
- Identify preliminary safety control structure
- Perform STPA

Perform safety-driven system design and analysis
- Make system-level design decisions to satisfy functional requirements and safety constraints
- Define component responsibilities
- Identify potentially unsafe control actions and restate as constraints on system and component behavior

Implementation (construction and manufacturing)

Document system limitations

Perform final safety assessment

Safety certification

Field testing, installation, and training

Operations, including maintenance and upgrades
- Change analysis
- Incident and accident analysis
- Performance monitoring
- Periodic audits

Decommissioning

Figure 10.3
System safety and human factors integrated into the set of typical system engineering tasks. Standard verification and validation activities are not shown as they are assumed to be performed throughout the whole process, not just at the end where they are often concentrated.

separation. Two types of advisories can be issued. *Resolution advisories* (RAs) provide instructions to the pilots to ensure safe separation from nearby traffic in the vertical plane.[1] *Traffic advisories* (TAs) indicate the positions of intruding aircraft that may later cause resolution advisories to be displayed.

TCAS is an example of a system created to directly impact safety where the goals are all directly related to safety. But system safety engineering and safety-driven design can be applied to systems where maintaining safety is not the only goal and, in fact, human safety is not even a factor. The example of an outer planets explorer spacecraft was shown in chapter 7. Another example is the air traffic control system, which has both safety and nonsafety (throughput) goals.

10.3.2 Defining Accidents

Before any safety-related activities can start, the definition of an accident needs to be agreed upon by the system customer and other stakeholders. This definition, in essence, establishes the goals for the safety effort.

Defining accidents in TCAS is straightforward—only one is relevant, a midair collision. Other more interesting examples are shown in chapter 7.

Basically, the criterion for specifying events as accidents is that the losses are so important that they need to play a central role in the design and tradeoff process. In the outer planets explorer example in chapter 7, some of the losses involve the mission goals themselves while others involve losses to other missions or a negative impact on our solar system ecology.

Priorities and evaluation criteria may be assigned to the accidents to indicate how conflicts are to be resolved, such as conflicts between safety goals or conflicts between mission goals and safety goals and to guide design choices at lower levels. The priorities are then inherited by the hazards related to each of the accidents and traced down to the safety-related design features.

10.3.3 Identifying the System Hazards

Once the set of accidents has been agreed upon, hazards can be derived from them. This process is part of what is called Preliminary Hazard Analysis (PHA) in System Safety. The hazard log is usually started as soon as the hazards to be considered are identified. While much of the information in the hazard log will be filled in later, some information is available at this time.

There is no right or wrong list of hazards—only an agreement by all involved on what hazards will be considered. Some hazards that were considered during the design of TCAS are listed in chapter 7 and are repeated here for convenience:

1. Horizontal advisories were originally planned for later versions of TCAS but have not yet been implemented.

1. TCAS causes or contributes to a near midair collision (NMAC), defined as a pair of controlled aircraft violating minimum separation standards.
2. TCAS causes or contributes to a controlled maneuver into the ground.
3. TCAS causes or contributes to the pilot losing control over the aircraft.
4. TCAS interferes with other safety-related aircraft systems (for example, ground proximity warning).
5. TCAS interferes with the ground-based air traffic control system (e.g., transponder transmissions to the ground or radar or radio services).
6. TCAS interferes with an ATC advisory that is safety-related (e.g., avoiding a restricted area or adverse weather conditions).

Once accidents and hazards have been identified, early concept formation (sometimes called high-level architecture development) can be started for the integrated system and safety engineering process.

10.3.4 Integrating Safety into Architecture Selection and System Trade Studies

An early activity in the system engineering of complex systems is the selection of an overall architecture for the system, or as it is sometimes called, system concept formation. For example, an architecture for manned space exploration might include a transportation system with parameters and options for each possible architectural feature related to technology, policy, and operations. Decisions will need to be made early, for example, about the number and type of vehicles and modules, the destinations for the vehicles, the roles and activities for each vehicle including dockings and undockings, trajectories, assembly of the vehicles (in space or on Earth), discarding of vehicles, prepositioning of vehicles in orbit and on the planet surface, and so on. Technology options include type of propulsion, level of autonomy, support systems (water and oxygen if the vehicle is used to transport humans), and many others. Policy and operational options may include crew size, level of international investment, types of missions and their duration, landing sites, and so on. Decisions about these overall system concepts clearly must precede the actual implementation of the system.

How are these decisions made? The selection process usually involves extensive tradeoff analysis that compares the different feasible architectures with respect to some important system property or properties. Cost, not surprisingly, usually plays a large role in the selection process while other properties, including system safety, are usually left as a problem to be addressed later in the development lifecycle. Many of the early architectural decisions, however, have a significant and lasting impact on safety and may not be reversible after the basic architectural decisions have been made. For example, the decision not to include a crew escape system on the Space Shuttle was an early architectural decision and has been impacting Shuttle safety for more than thirty years [74, 136]. After the *Challenger* accident and again after the *Columbia* loss, the idea resurfaced, but there was no cost-effective way to add crew escape at that time.

The primary reason why safety is rarely factored in during the early architectural tradeoff process, except perhaps informally, is that practical methods for analyzing safety, that is, hazard analysis methods that can be applied at that time, do not exist. But if information about safety were available early, it could be used in the selection process and hazards could be eliminated by the selection of appropriate architectural options or mitigated early when the cost of doing so is much less than later in the system lifecycle. Making basic design changes downstream becomes increasingly costly and disruptive as development progresses and, often, compromises in safety must be accepted that could have been eliminated if safety had been considered in the early architectural evaluation process.

While it is relatively easy to identify hazards at system conception, performing a hazard or risk assessment before a design is available is more problematic. At best, only a very rough estimate is possible. Risk is usually defined as a combination of severity and likelihood. Because these two different qualities (severity and likelihood) cannot be combined mathematically, they are commonly qualitatively combined using a risk matrix. Figure 10.4 shows a fairly standard form for such a matrix.

		SEVERITY			
		I Catastrophic	II Critical	III Marginal	IV Negligible
LIKELIHOOD	A Frequent	I–A	II–A	III–A	IV–A
	B Moderate	I–B	II–B	III–B	IV–B
	C Occasional	I–C	II–C	III–C	IV–C
	D Remote	I–D	II–D	III–D	IV–D
	E Unlikely	I–E	II–E	III–E	IV–E
	F Impossible	I–F	II–F	III–F	IV–F

Figure 10.4
A standard risk matrix.

High-level hazards are first identified and, for each identified hazard, a qualitative evaluation is performed by classifying the hazard according to its severity and likelihood.

While severity can usually be evaluated using the worst possible consequences of that hazard, likelihood is almost always unknown and, arguably, unknowable for complex systems before any system design decisions have been made. The problem is even worse before a system architecture has been selected. Some probabilistic information is usually available about physical events, of course, and historical information may theoretically be available. But new systems are usually being created because existing systems and designs are not adequate to achieve the system goals, and the new systems will probably use new technology and design features that limit the accuracy of historical information. For example, historical information about the likelihood of propulsion-related losses may not be accurate for new spacecraft designs using nuclear propulsion. Similarly, historical information about the errors air traffic controllers make has no relevance for new air traffic control systems, where the type of errors may change dramatically.

The increasing use of software in most complex systems complicates the situation further. Much or even most of the software in the system will be new and have no historical usage information. In addition, statistical techniques that assume randomness are not applicable to software design flaws. Software and digital systems also introduce new ways for hazards to occur, including new types of component interaction accidents. Safety is a system property, and, as argued in part I, combining the probability of failure of the system components to be used has little or no relationship to the safety of the system as a whole.

There are no known or accepted rigorous or scientific ways to obtain probabilistic or even subjective likelihood information using historical data or analysis in the case of non-random failures and system design errors, including unsafe software behavior. When forced to come up with such evaluations, engineering judgment is usually used, which in most cases amounts to pulling numbers out of the air, often influenced by political and other nontechnical factors. Selection of a system architecture and early architectural trade evaluations on such a basis is questionable and perhaps one reason why risk usually does not play a primary role in the early architectural trade process.

Alternatives to the standard risk matrix are possible, but they tend to be application specific and so must be constructed for each new system. For many systems, the use of severity alone is often adequate to categorize the hazards in trade studies. Two examples of other alternatives are presented here, one created for augmented air traffic control technology and the other created and used in the early architectural trade study of NASA's Project Constellation, the program to return to the moon and later go on to Mars. The reader is encouraged to come up with their own methods appropriate for their particular application. The examples are not meant to be definitive, but simply illustrative of what is possible.

Example 1: A Human-Intensive System: Air Traffic Control Enhancements

Enhancements to the air traffic control (ATC) system are unique in that the problem is not to create a new or safer system but to maintain the very high level of safety built into the current system: The goal is to not degrade safety. The risk likelihood estimate can be restated, in this case, as the likelihood that safety will be degraded by the proposed changes and new tools. To tackle this problem, we created a set of criteria to be used in the evaluation of likelihood.[2] The criteria ranked various high-level architectural design features of the proposed set of ATC tools on a variety of factors related to risk in these systems. The ranking was qualitative and most criteria were ranked as having low, medium, or high impact on the likelihood of safety being degraded from the current level. For the majority of factors, "low" meant insignificant or no change in safety with respect to that factor in the new versus the current system, "medium" denoted the potential for a minor change, and "high" signified potential for a significant change in safety. Many of the criteria involve human-automation interaction, since ATC is a very human-intensive system and the new features being proposed involved primarily new automation to assist human air traffic controllers. Here are examples of the likelihood level criteria used:

- *Safety margins:* Does the new feature have the potential for (1) an insignificant or no change to the existing safety margins, (2) a minor change, or (3) a significant change.
- *Situation awareness:* What is the level of change in the potential for reducing situation awareness.
- *Skills currently used and those necessary to backup and monitor the new decision-support tools:* Is there an insignificant or no change in the controller skills, a minor change, or a significant change.
- *Introduction of new failure modes and hazard causes:* Do the new tools have the same function and failure modes as the system components they are replacing, are new failure modes and hazards introduced but well understood and effective mitigation measures can be designed, or are the new failure modes and hazard causes difficult to control.
- *Effect of the new software functions on the current system hazard mitigation measures:* Can the new features render the current safety measures ineffective or are they unrelated to current safety features.

2. These criteria were developed for a NASA contract by the author and have not been published previously.

- *Need for new system hazard mitigation measures:* Will the proposed changes require new hazard mitigation measures.

These criteria and others were converted into a numerical scheme so they could be combined and used in an early risk assessment of the changes being contemplated and their potential likelihood for introducing significant new risk into the system. The criteria were weighted to reflect their relative importance in the risk analysis.

Example 2: Early Risk Analysis of Manned Space Exploration

A second example was created by Nicolas Dulac and others as part of an MIT and Draper Labs contract with NASA to perform an architectural tradeoff analysis for future human space exploration [59]. The system engineers wanted to include safety along with the usual factors, such as mass, to evaluate the candidate architectures, but once again little information was available at this early stage of system engineering. It was not possible to evaluate likelihood using historical information; all of the potential architectures involved new technology, new missions, and significant amounts of software.

In the procedure developed to achieve the goal, the hazards were first identified as shown in figure 10.5. As is the case at the beginning of any project, identifying system hazards involved ten percent creativity and ninety percent experience. Hazards were identified for each mission phase by domain experts under the guidance of the safety experts. Some hazards, such as fire, explosion, or loss of life-support span multiple (if not all) mission phases and were grouped as *General Hazards*. The control strategies used to mitigate them, however, may depend on the mission phase in which they occur.

Once the hazards were identified, the severity of each hazard was evaluated by considering the worst-case loss associated with the hazard. In the example, the losses are evaluated for each of three categories: humans (H), mission (M), and equipment (E). Initially, potential damage to the Earth and planet surface environment was included in the hazard log. In the end, the environment component was left out of the analysis because project managers decided to replace the analysis with mandatory compliance with NASA's planetary protection standards. A risk analysis can be replaced by a customer policy on how the hazards are to be treated. A more complete example, however, for a different system would normally include environmental hazards.

A severity scale was created to account for the losses associated with each of the three categories. The scale used is shown in figure 10.6, but obviously a different scale could easily be created to match the specific policies or standard practice in different industries and companies.

As usual, severity was relatively easy to handle but the likelihood of the potential hazard occurring was unknowable at this early stage of system engineering. In

ID#	Phase	Hazard	Severity		
			H	M	E
G1	General	Flammable substance in presence of ignition source (Fire)	4	4	4
G2	General	Flammable substance in presence of ignition source in confined space (explosion)	4	4	4
G3	General	Loss of life support (includes power, temperature, oxygen, air pressure, food, water, ...)	4	4	4
G4	General	Crew injury or illness	4	4	1
G5	General	Solar or nuclear radiation exceeding safe levels	3	3	2
G6	General	Collision (with micrometeroids, debris, during rendezvous or separation maneuver, ...)	4	4	4
G7	General	Loss of attitude control	4	4	4
G8	General	Engines do not ignite	4	4	2
PL1	Pre Launch	Damage to payload	2	3	3
PL2	Pre Launch	Launch delay (due to weather, pre launch test failues, etc.)	1	4	1
L1	Launch	Incorrect propulsion/trajectory/control during ascent	4	4	4
L2	Launch	Loss of structural integrity (due to aerodynamic loads, vibrations, ...)	4	4	4
L3	Launch	Incorrect stage separation	4	4	4
E1	EVA in Space	Astronaut lost in space	4	4	1
AS1	Assembly	Incorrect propulsion/control during rendezvous	4	4	4
AS2	Assembly	Inability to dock	1	4	3
AS3	Assembly	Inability to achieve airlock during docking	1	4	3
AS4	Assembly	Inability to undock	4	4	3
T1	Course Change	Incorrect propulsion/trajectory/control during course change burn	4	4	3
D1	Descent	Inability to undock	4	4	3
D2	Descent	Incorrect propulsion/trajectory/control during descent	4	4	4
D3	Descent	Loss of structural integrity (due to inadequate thermal control, aerodynamic loads, vibrations, ...)	4	4	4
AC1	Ascent	Incorrect stage separation (including ascent module disconnecting from descent stage)	4	3	3
AC2	Ascent	Incorrect propulsion/trajectory/control during ascent	4	3	3
AC3	Ascent	Loss of structural integrity (due to aerodynamic loads, vibrations, ...)	4	3	3
S1	Surface Ops	Crew members stranded on Moon/Mars surface during EVA (Extra Vehicle Activity)	4	3	3
S2	Surface Ops	Crew members lost on Moon/Mars surface during EVA	4	3	3
S3	Surface Ops	Equipment damage (includingdamage related to lunar dust)	2	3	3
NP1	Nuclear Power	Nuclear fuel released on Earth surface	4	4	2
NP2	Nuclear Power	Insufficient power generation (reactor does not work)	4	3	3
NP3	Nuclear Power	Insufficient reactor cooling (leading to reactor meltdown)	4	3	3
RE1	Reentry	Inability to undock	4	3	3
RE2	Reentry	Incorrect propulsion/trajectory/control during descent	4	3	3
RE3	Reentry	Loss of structural integrity (due to inadequate thermal control, aerodynamic loads, vibrations, ...)	4	3	4

Figure 10.5
System-level hazards and associated severities.

Severity Level	Human	Mission	Equipment
4	Loss of Life	Mission abort or mission loss	System loss
3	Severe Injury or Illness	Major mission objectives incomplete	Major system damage
2	Minor Injury or illness	Minor mission objectives incomplete	Minor system damage
1	No /insignificant injury or illness	All mission objectives completed	No/insignificant damage

Figure 10.6
Custom severity scale for the candidate architectures analysis.

addition, space exploration is the polar opposite of the ATC example above as the system did not already exist and the architectures and missions would involve things never attempted before, which created a need for a different approach to estimating likelihood.

We decided to use the *mitigation potential* of the hazard in the candidate architecture as an estimator of, or surrogate for, likelihood. Hazards that are more easily mitigated in the design and operations are less likely to lead to accidents. Similarly, hazards that have been eliminated during system design, and thus are not part of that candidate architecture or can easily be eliminated in the detailed design process, cannot lead to an accident.

The safety goal of the architectural analysis process was to assist in selecting the architecture with the fewest serious hazards and highest mitigation potential for those hazards that were not eliminated. Not all hazards will be eliminated even if they can be. One reason for not eliminating hazards might be that it would reduce the potential for achieving other important system goals or constraints. Obviously, safety is not the only consideration in the architecture selection process, but it is important enough in this case to be a criterion in the selection process.

Mitigation potential was chosen as a surrogate for likelihood for two reasons: (1) the potential for eliminating or controlling the hazard in the design or operations has a direct and important bearing on the likelihood of the hazard occurring (whether traditional or new designs and technology are used) and (2) mitigatibility of the hazard can be determined before an architecture or design is selected—indeed, it assists in the selection process.

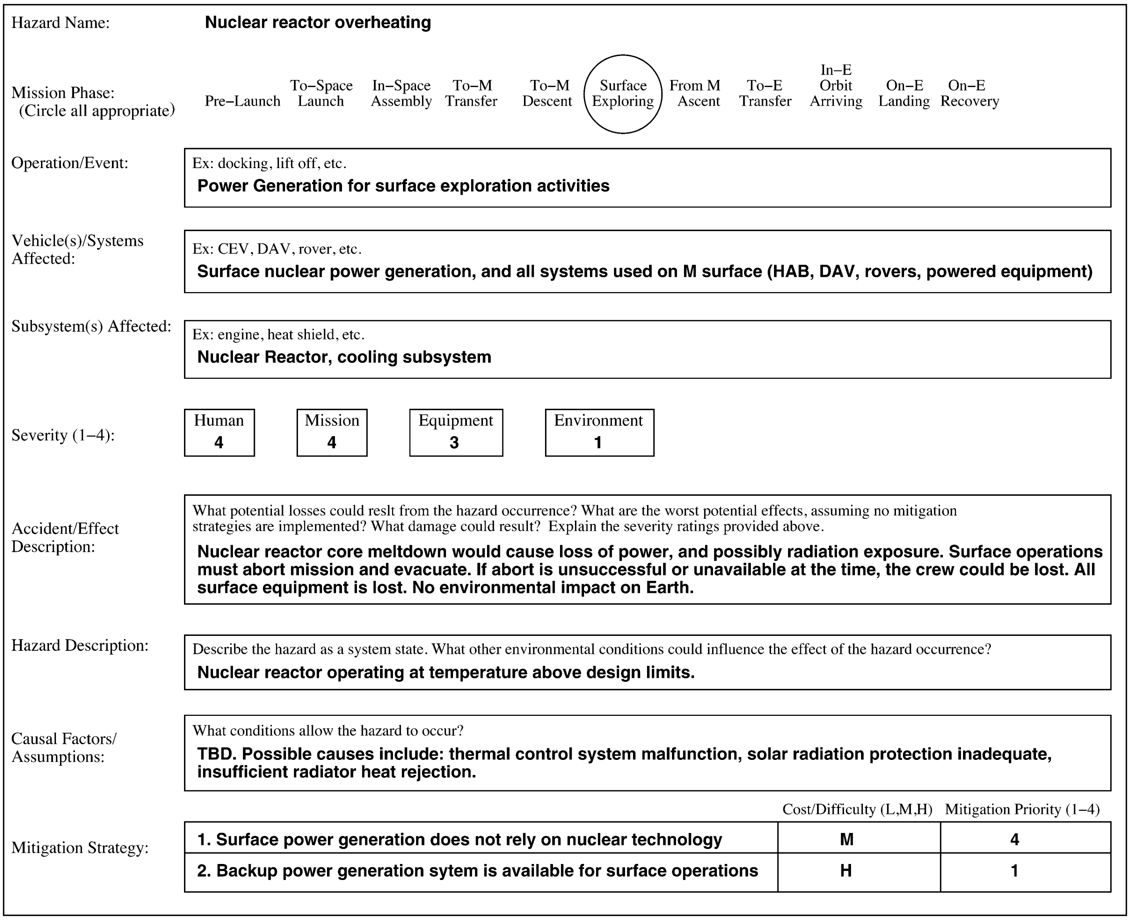
Figure 10.7 shows an example from the hazard log created during the PHA effort. The example hazard shown is *nuclear reactor overheating*. Nuclear power generation and use, particularly during planetary surface operations, was considered to be an important option in the architectural tradeoffs. The potential accident and its effects are described in the hazard log as:

> Nuclear core meltdown would cause loss of power, and possibly radiation exposure. Surface operations must abort mission and evacuate. If abort is unsuccessful or unavailable at the time, the crew and surface equipment could be lost. There would be no environmental impact on Earth.

The hazard is defined as the nuclear reactor operating at temperatures above the design limits.

Although some causal factors can be hypothesized early, a hazard analysis using STPA can be used to generate a more complete list of causal factors later in the development process to guide the design process after an architecture is chosen.

Like severity, mitigatibility was evaluated by domain experts under the guidance of safety experts. Both the cost of the potential mitigation strategy and its



Hazard Name: **Nuclear reactor overheating**

Mission Phase: (Circle all appropriate) Pre–Launch | To–Space Launch | In–Space Assembly | To–M Transfer | To–M Descent | [✓ Surface Exploring] | From M Ascent | To–E Transfer | In–E Orbit Arriving | On–E Landing | On–E Recovery

Operation/Event: Ex: docking, lift off, etc.
Power Generation for surface exploration activities

Vehicle(s)/Systems Affected: Ex: CEV, DAV, rover, etc.
Surface nuclear power generation, and all systems used on M surface (HAB, DAV, rovers, powered equipment)

Subsystem(s) Affected: Ex: engine, heat shield, etc.
Nuclear Reactor, cooling subsystem

Severity (1–4):

Human	Mission	Equipment	Environment
4	**4**	**3**	**1**

Accident/Effect Description: What potential losses could reslt from the hazard occurrence? What are the worst potential effects, assuming no mitigation strategies are implemented? What damage could result? Explain the severity ratings provided above.
Nuclear reactor core meltdown would cause loss of power, and possibly radiation exposure. Surface operations must abort mission and evacuate. If abort is unsuccessful or unavailable at the time, the crew could be lost. All surface equipment is lost. No environmental impact on Earth.

Hazard Description: Describe the hazard as a system state. What other environmental conditions could influence the effect of the hazard occurrence?
Nuclear reactor operating at temperature above design limits.

Causal Factors/ Assumptions: What conditions allow the hazard to occur?
TBD. Possible causes include: thermal control system malfunction, solar radiation protection inadequate, insufficient radiator heat rejection.

Mitigation Strategy:

	Cost/Difficulty (L,M,H)	Mitigation Priority (1–4)
1. Surface power generation does not rely on nuclear technology	**M**	**4**
2. Backup power generation sytem is available for surface operations	**H**	**1**

Figure 10.7
A sample from the hazard log generated during the preliminary hazard analysis for the space architecture candidate tradeoff analysis.

Level	General Description	Detailed Description
4	Eliminate	Complete elimination of the hazard from the design
3	Prevent	Reduction of the likelihood that the hazard will occur
2	Control	Reduction of the likelihood that the hazard results in an accident
1	Reduce Damage	Reduction of damage if an accident does occur

Figure 10.8
A sample hazard-mitigation priority scale.

effectiveness were evaluated. For the nuclear power example, two strategies were identified: the first is not to use nuclear power generation at all. The cost of this option was evaluated as medium (on a low, medium, high scale). But the mitigation potential was rated as high because it eliminates the hazard completely. The mitigation priority scale used is shown in figure 10.8. The second mitigation potential identified by the engineers was to provide a backup power generation system for surface operations. The difficulty and cost was rated high and the mitigation rating was 1, which was the lowest possible level, because at best it would only reduce the damage if an accident occurred but potential serious losses would still occur. Other mitigation strategies are also possible but have been omitted from the sample hazard log entry shown.

None of the effort expended here is wasted. The information included in the hazard log about the mitigation strategies will be useful later in the design process if the final architecture selected uses surface nuclear power generation. NASA might also be able to use the information in future projects and the creation of such early risk analysis information might be common to companies or industries and not have to be created for each project. As new technologies are introduced to an industry, new hazards or mitigation possibilities could be added to the previously stored information.

The final step in the process is to create safety risk metrics for each candidate architecture. Because the system engineers on the project created hundreds of feasible architectures, the evaluation process was automated. The actual details of the mathematical procedures used are of limited general interest and are available elsewhere [59]. Weighted averages were used to combine mitigation factors and severity factors to come up with a final *Overall Residual Safety-Risk Metric*. This metric was then used in the evaluation and ranking of the potential manned space exploration architectures.

By selecting and deselecting options in the architecture description, it was also possible to perform a first-order assessment of the relative importance of each architectural option in determining the Overall Residual Safety-Risk Metric.

While hundreds of parameters were considered in the risk analysis, the process allowed the identification of major contributors to the hazard mitigation potential of selected architectures and thus informed the architecture selection process and the tradeoff analysis. For example, important contributors to increased safety were determined to include the use of heavy module and equipment prepositioning on the surface of Mars and the use of minimal rendezvous and docking maneuvers. Prepositioning modules allows for pretesting and mitigates the hazards associated with loss of life support, equipment damage, and so on. On the other hand, prepositioning modules increases the reliance on precision landing to ensure that all landed modules are within range of each other. Consequently, using heavy prepositioning may require additional mitigation strategies and technology development to reduce the risk associated with landing in the wrong location. All of this information must be considered in selecting the best architecture. As another example, on one hand, a transportation architecture requiring no docking at Mars orbit or upon return to Earth inherently mitigates hazards associated with collisions or failed rendezvous and docking maneuvers. On the other hand, having the capability to dock during an emergency, even though it is not required during nominal operations, provides additional mitigation potential for loss of life support, especially in Earth orbit.

Reducing these considerations to a number is clearly not ideal, but with hundreds of potential architectures it was necessary in this case in order to pare down the choices to a smaller number. More careful tradeoff analysis is then possible on the reduced set of choices.

While mitigatibility is widely applicable as a surrogate for likelihood in many types of domains, the actual process used above is just one example of how it might be used. Engineers will need to adapt the scales and other features of the process to the customary practices in their own industry. Other types of surrogates or ways to handle likelihood estimates in early phases of projects are possible beyond the two examples provided in this section. While none of these approaches is ideal, they are much better than ignoring safety in decision making or selecting likelihood estimates based solely on wishful thinking or the politics that often surround the preliminary hazard analysis process.

After a conceptual design is chosen, development begins.

10.3.5 Documenting Environmental Assumptions

An important part of the system development process is to determine and document the assumptions under which the system requirements and design features are derived and upon which the hazard analysis is based. Assumptions will be identified and specified throughout the system engineering process and the engineering specifications to explain decisions or to record fundamental information upon which the design is based. If the assumptions change over time or the system changes and the assumptions are no longer true, then the requirements and the safety constraints and design features based on those assumptions need to be revisited to ensure safety has not been compromised by the change.

Because operational safety depends on the accuracy of the assumptions and models underlying the design and hazard analysis processes, the operational system should be monitored to ensure that:

1. The system is constructed, operated, and maintained in the manner assumed by the designers.
2. The models and assumptions used during initial decision making and design are correct.
3. The models and assumptions are not violated by changes in the system, such as workarounds or unauthorized changes in procedures, or by changes in the environment.

Operational feedback on trends, incidents, and accidents should trigger reanalysis when appropriate. Linking the assumptions throughout the document with the parts of the hazard analysis based on that assumption will assist in performing safety maintenance activities.

Several types of assumptions are relevant. One is the assumptions under which the system will be used and the environment in which the system will operate. Not only will these assumptions play an important role in system development, but they also provide part of the basis for creating the operational safety control structure and other operational safety controls such as creating feedback loops to ensure the assumptions underlying the system design and the safety analyses are not violated during operations as the system and its environment change over time.

While many of the assumptions that originate in the existing environment into which the new system will be integrated can be identified at the beginning of development, additional assumptions will be identified as the design process continues and new requirements and design decisions and features are identified. In addition, assumptions that the emerging system design imposes on the surrounding environment will become clear only after detailed decisions are made in the design and safety analyses.

Examples of important environment assumptions for TCAS II are that:

EA1: *High-integrity communications exist between aircraft.*

EA2: *The TCAS-equipped aircraft carries a Mode-S air traffic control transponder.*[3]

3. An aircraft *transponder* sends information to help air traffic control maintain aircraft separation. Primary radar generally provides bearing and range position information, but lacks altitude information. Mode A transponders transmit only an identification signal, while Mode C and Mode S transponders also report pressure altitude. Mode S is newer and has more capabilities than Mode C, some of which are required for the collision avoidance functions in TCAS.

EA3: *All aircraft have operating transponders.j*

EA4: *All aircraft have legal identification numbers.*

EA5: *Altitude information is available from intruding targets with a minimum precision of 100 feet.*

EA6: *The altimetry system that provides own aircraft pressure altitude to the TCAS equipment will satisfy the requirements in RTCA Standard . . .*

EA7: *Threat aircraft will not make an abrupt maneuver that thwarts the TCAS escape maneuver.*

As noted, these assumptions must be enforced in the overall safety control structure. With respect to assumption EA4, for example, identification numbers are usually provided by the aviation authorities in each country, and that requirement will need to be ensured by international agreement or by some international agency. The assumption that aircraft have operating transponders (EA3) may be enforced by the airspace rules in a particular country and, again, must be ensured by some group. Clearly, these assumptions play an important role in the construction of the safety control structure and assignments of responsibilities for the final system. For TCAS, some of these assumptions will already be imposed by the existing air transportation safety control structure while others may need to be added to the responsibilities of some group(s) in the control structure. The last assumption, EA7, imposes constraints on pilots and the air traffic control system.

Environment requirements and constraints may lead to restrictions on the use of the new system (in this case, TCAS) or may indicate the need for system safety and other analyses to determine the constraints that must be imposed on the system being created (TCAS again) or the larger encompassing system to ensure safety. The requirements for the integration of the new subsystem safely into the larger system must be determined early. Examples for TCAS include:

E1: *The behavior or interaction of non-TCAS equipment with TCAS must not degrade the performance of the TCAS equipment or the performance of the equipment with which TCAS interacts.*

E2: *Among the aircraft environmental alerts, the hierarchy shall be: Windshear has first priority, then the Ground Proximity Warning System (GPWS), then TCAS.*

E3: *The TCAS alerts and advisories must be independent of those using the master caution and warming system.*

10.3.6 System-Level Requirements Generation

Once the goals and hazards have been identified and a conceptual system architecture has been selected, system-level requirements generation can begin. Usually, in the early stages of a project, goals are stated in very general terms, as shown in G1 and G2. One of the first steps in the design process is to refine the goals into testable and achievable high-level requirements (the "shall" statements). Examples of high-level functional requirements implementing the goals for TCAS are:

1.18: *TCAS shall provide collision avoidance protection for any two aircraft closing horizontally at any rate up to 1200 knots and vertically up to 10,000 feet per minute.*

Assumption: *This requirement is derived from the assumption that commercial aircraft can operate up to 600 knots and 5000 fpm during vertical climb or controlled descent (and therefore two planes can close horizontally up to 1200 knots and vertically up to 10,000 fpm).*

1.19.1: *TCAS shall operate in enroute and terminal areas with traffic densities up to 0.3 aircraft per square nautical miles (i.e., 24 aircraft within 5 nmi).*

Assumption: *Traffic density may increase to this level by 1990, and this will be the maximum density over the next 20 years.*

As stated earlier, *assumptions* should continue to be specified when appropriate to explain a decision or to record fundamental information on which the design is based. Assumptions are an important component of the documentation of design rationale and form the basis for safety audits during operations. Consider the above requirement labeled 1.18, for example. In the future, if aircraft performance limits change or there are proposed changes in airspace management, the origin of the specific numbers in the requirement (1,200 and 10,000) can be determined and evaluated for their continued relevance. In the absence of the documentation of such assumptions and how they impact the detailed design decisions, numbers tend to become "gospel," and everyone is afraid to change them.

Requirements (and constraints) must also be included for the human operator and for the human–computer interface. These requirements will in part be derived from the *concept of operations*, which should in turn include a *human task analysis* [48, 47], to determine how TCAS is expected to be used by pilots (which, again, should be checked in safety audits during operations). These analyses use information about the goals of the system, the constraints on how the goals are achieved, including safety constraints, how the automation will be used, how humans now control the system and work in the system without automation, and the tasks humans need to perform and how the automation will support them in performing these tasks. The task analysis must also consider workload and its impact on operator performance. Note that a low workload may be more dangerous than a high one.

Requirements on the operator (in this case, the pilot) are used to guide the design of the TCAS-pilot interface, the design of the automation logic, flight-crew tasks and procedures, aircraft flight manuals, and training plans and program. Traceability links should be provided to show the relationships. Links should also be provided to the parts of the hazard analysis from which safety-related requirements are derived. Examples of TCAS II operator safety requirements and constraints are:

OP.4: *After the threat is resolved, the pilot shall return promptly and smoothly to his/her previously assigned fight path (→ HA-560, ↓3.3).*

OP.9: *The pilot must not maneuver on the basis of a Traffic Advisory only (→ HA-630, ↓2.71.3).*

The requirements and constraints include links to the hazard analysis that produced the information and to design documents and decisions to show where the requirements are applied. These two examples have links to the parts of the hazard analysis from which they were derived, links to the system design and operator procedures where they are enforced, and links to the user manuals (in this case, the pilot manuals) to explain why certain activities or behaviors are required.

The links not only provide traceability from requirements to implementation and vice versa to assist in review activities, but they also embed the design rationale information into the specification. If changes need to be made to the system, it is easy to follow the links and determine why and how particular design decisions were made.

10.3.7 Identifying High-Level Design and Safety Constraints

Design constraints are restrictions on how the system can achieve its purpose. For example, TCAS is not allowed to interfere with the ground-level air traffic control system while it is trying to maintain adequate separation between aircraft. Avoiding interference is not a goal or purpose of TCAS—the best way to achieve the goal is not to build the system at all. It is instead a constraint on how the system can achieve its purpose, that is, a constraint on the potential system designs. Because of the need to evaluate and clarify tradeoffs among alternative designs, separating these two types of intent information (goals and design constraints) is important.

For safety-critical systems, constraints should be further separated into safety-related and not safety-related. One nonsafety constraint identified for TCAS, for example, was that requirements for new hardware and equipment on the aircraft be minimized or the airlines would not be able to afford this new collision avoidance system. Examples of nonsafety constraints for TCAS II are:

C.1: *The system must use the transponders routinely carried by aircraft for ground ATC purposes (↓2.3, 2.6).*

Rationale: *To be acceptable to airlines, TCAS must minimize the amount of new hardware needed.*

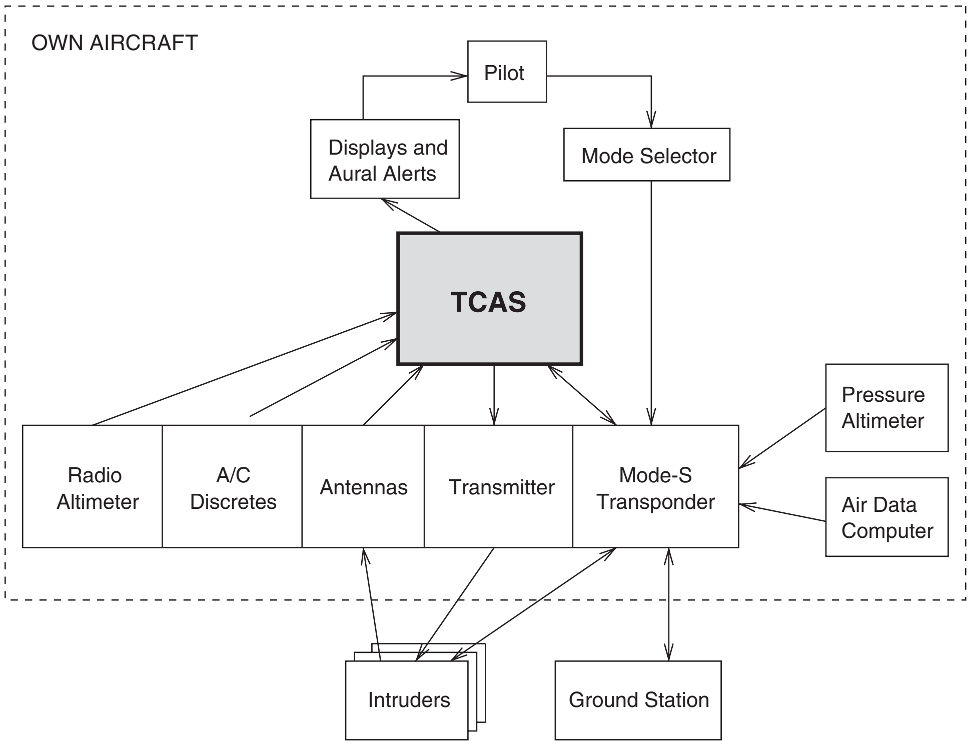


Figure 10.9
The system interface topology for TCAS.

C.4: *TCAS must comply with all applicable FAA and FCC policies, rules, and philosophies* (↓2.30, 2.79).

The physical environment with which TCAS interacts is shown in figure 10.9. The constraints imposed by these existing environmental components must also be identified before system design can begin.

Safety-related constraints should have two-way links to the system hazard log and to any analysis results that led to that constraint being identified as well as links to the design features (usually level 2) included to eliminate or control them. Hazard analyses are linked to level 1 requirements and constraints, to design features on level 2, and to system limitations (or accepted risks). An example of a level 1 safety constraint derived to prevent hazards is:

SC.3: *TCAS must generate advisories that require as little deviation as possible from ATC clearances* (→ H6, HA-550, ↓2.30).

The link in SC.3 to 2.30 points to the level 2 system design feature that implements this safety constraint. The other links provide traceability to the hazard (H6) from which the constraint was derived and to the parts of the hazard analysis involved, in this case the part of the hazard analysis labeled HA-550.

The following is another example of a safety constraint for TCAS II and some constraints refined from it, all of which stem from a high-level environmental constraint derived from safety considerations in the encompassing system into which TCAS will be integrated. The refinement will occur as safety-related decisions are made and guided by an STPA hazard analysis:

SC.2: *TCAS must not interfere with the ground ATC system or other aircraft transmissions to the ground ATC system (→ H5).*

SC.2.1: *The system design must limit interference with ground-based secondary surveillance radar, distance-measuring equipment channels, and with other radio services that operate in the 1030/1090 MHz frequency band (↓2.5.1).*

SC.2.1.1: *The design of the Mode S waveforms used by TCAS must provide compatibility with Modes A and C of the ground-based secondary surveillance radar system (↓2.6).*

SC.2.1.2: *The frequency spectrum of Mode S transmissions must be controlled to protect adjacent distance-measuring equipment channels (↓2.13).*

SC.2.1.3: *The design must ensure electromagnetic compatibility between TCAS and [...] [↓21.4).*

SC.2.2: *Multiple TCAS units within detection range of one another (approximately 30 nmi) must be designed to limit their own transmissions. As the number of such TCAS units within this region increases, the interrogation rate and power allocation for each of them must decrease in order to prevent undesired interference with ATC (↓2.13).*

Assumptions are also associated with safety constraints. As an example of such an assumption, consider:

SC.6: *TCAS must not disrupt the pilot and ATC operations during critical phases of flight nor disrupt aircraft operation (→ H3, ↓2.2.3, 2.19, 2.24.2).*

SC.6.1: *The pilot of a TCAS-equipped aircraft must have the option to switch to the Traffic-Advisory-Only mode where TAs are displayed but display of resolution advisories is inhibited (↓ 2.2.3).*

Assumption: *This feature will be used during final approach to parallel runways, when two aircraft are projected to come close to each other and TCAS would call for an evasive maneuver (↓ 6.17).*

The specified assumption is critical for evaluating safety during operations. Humans tend to change their behavior over time and use automation in different ways than originally intended by the designers. Sometimes, these new uses are dangerous. The hyperlink at the end of the assumption (↓ 6.17) points to the required auditing procedures for safety during operations and to where the procedures for auditing this assumption are specified.

Where do these safety constraints come from? Is the system engineer required to simply make them up? While domain knowledge and expertise is always going to be required, there are procedures that can be used to guide this process.

The highest-level safety constraints come directly from the identified hazards for the system. For example, TCAS must not cause or contribute to a near miss (H1), TCAS must not cause or contribute to a controlled maneuver into the ground (H2), and TCAS must not interfere with the ground-based ATC system. STPA can be used to refine these high-level design constraints into more detailed design constraints as described in chapter 8.

The first step in STPA is to create the high-level TCAS operational safety control structure. For TCAS, this structure is shown in figure 10.10. For simplicity, much of the structure above ATC operations management has been omitted and the roles and responsibilities have been simplified here. In a real design project, roles and responsibilities will be augmented and refined as development proceeds, analyses are performed, and design decisions are made. Early in the system concept formation, specific roles may not all have been determined, and more will be added as the design concepts are refined. One thing to note is that there are three groups with potential responsibilities over the pilot's response to a potential NMAC: TCAS, the ground ATC, and the airline operations center which provides the airline procedures for responding to TCAS alerts. Clearly any potential conflicts and coordination problems between these three controllers will need to be resolved in the overall air traffic management system design. In the case of TCAS, the designers decided that because there was no practical way, at that time, to downlink information to the ground controllers about any TCAS advisories that might have been issued for the crew, the pilot was to immediately implement the TCAS advisory and the co-pilot would transmit the TCAS alert information by radio to ground ATC. The airline would provide the appropriate procedures and training to implement this protocol.

Part of defining this control structure involves identifying the responsibilities of each of the components related to the goal of the system, in this case collision avoidance. For TCAS, these responsibilities include:

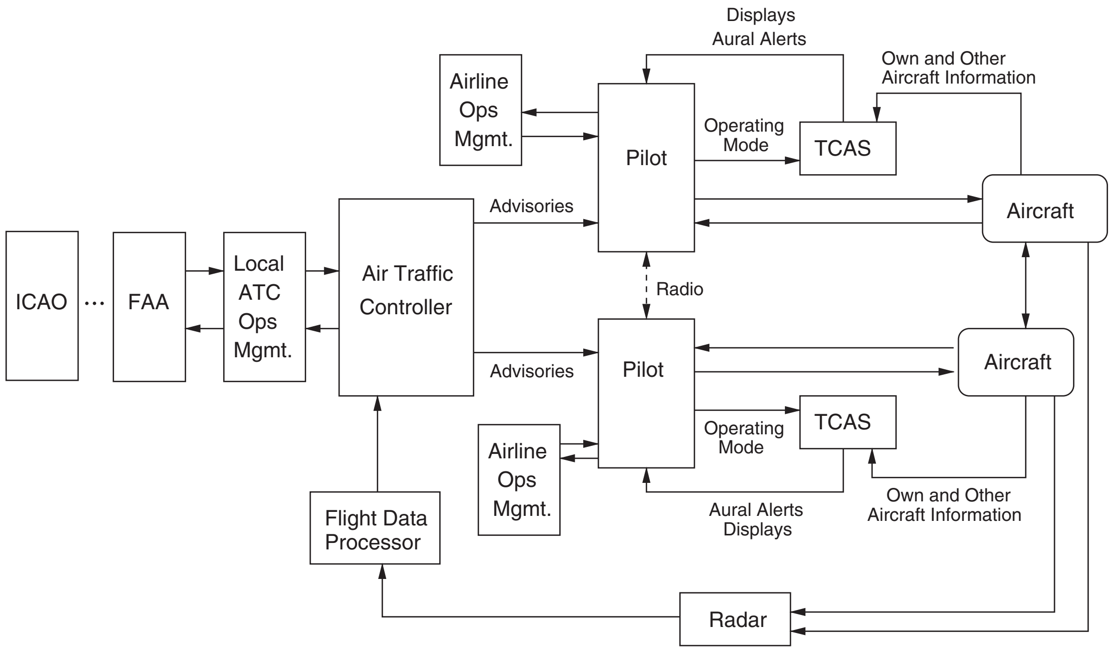


Figure 10.10
The high-level operational TCAS control structure.

- *Aircraft Components* (e.g., transponders, antennas): Execute control maneuvers, read and send messages to other aircraft, etc.
- *TCAS:* Receive information about its own and other aircraft, analyze the information received and provide the pilot with (1) information about where other aircraft in the vicinity are located and (2) an escape maneuver to avoid potential NMAC threats.
- *Aircraft Components* (e.g., transponders, antennas): Execute pilot-generated TCAS control maneuvers, read and send messages to and from other aircraft, etc.
- *Pilot:* Maintain separation between own and other aircraft, monitor the TCAS displays, and implement TCAS escape maneuvers. The pilot must also follow ATC advisories.
- *Air Traffic Control:* Maintain separation between aircraft in the controlled airspace by providing advisories (control actions) for the pilot to follow. TCAS is designed to be independent of and a backup for the air traffic controller so ATC does not have a direct role in the TCAS safety control structure but clearly has an indirect one.

- *Airline Operations Management:* Provide procedures for using TCAS and following TCAS advisories, train pilots, and audit pilot performance.
- *ATC Operations Management:* Provide procedures, train controllers, audit performance of controllers and of the overall collision avoidance system.
- *ICAO:* Provide worldwide procedures and policies for the use of TCAS and provide oversight that each country is implementing them.

After the general control structure has been defined (or alternative candidate control structures identified), the next step is to determine how the controlled system (the two aircraft) can get into a hazardous state. That information will be used to generate safety constraints for the designers. STAMP assumes that hazardous states (states that violate the safety constraints) are the result of ineffective control. Step 1 of STPA is to identify the potentially inadequate control actions.

Control actions in TCAS are called resolution advisories or RAs. An RA is an aircraft escape maneuver created by TCAS for the pilots to follow. Example resolution advisories are DESCEND, INCREASE RATE OF CLIMB TO 2500 FMP, and DON'T DESCEND. Consider the TCAS component of the control structure (see figure 10.10) and the NMAC hazard. The four types of control flaws for this example translate into:

1. The aircraft are on a near collision course, and TCAS does not provide an RA that avoids it (that is, does not provide an RA, or provides an RA that does not avoid the NMAC).
2. The aircraft are in close proximity and TCAS provides an RA that degrades vertical separation (causes an NMAC).
3. The aircraft are on a near collision course and TCAS provides a maneuver too late to avoid an NMAC.
4. TCAS removes an RA too soon.

These inadequate control actions can be restated as high-level constraints on the behavior of TCAS:

1. TCAS must provide resolution advisories that avoid near midair collisions.
2. TCAS must not provide resolution advisories that degrade vertical separation between two aircraft (that is, cause an NMAC).
3. TCAS must provide the resolution advisory while enough time remains for the pilot to avoid an NMAC. (A human factors and aerodynamic analysis should be performed at this point to determine exactly how much time that implies.)
4. TCAS must not remove the resolution advisory before the NMAC is resolved.

Similarly, for the pilot, the inadequate control actions are:

1. The pilot does not provide a control action to avoid a near midair collision.
2. The pilot provides a control action that does not avoid the NMAC.
3. The pilot provides a control action that causes an NMAC that would not otherwise have occurred.
4. The pilot provides a control action that could have avoided the NMAC but it was too late.
5. The pilot starts a control action to avoid an NMAC but stops it too soon.

Again, these inadequate pilot control actions can be restated as safety constraints that can be used to generate pilot procedures. Similar hazardous control actions and constraints must be identified for each of the other system components. In addition, inadequate control actions must be identified for the other functions provided by TCAS (beyond RAs) such as traffic advisories.

Once the high-level design constraints have been identified, they must be refined into more detailed design constraints to guide the system design and then augmented with new constraints as design decisions are made, creating a seamless integrated and iterative process of system design and hazard analysis.

Refinement of the constraints involves determining how they could be violated. The refined constraints will be used to guide attempts to eliminate or control the hazards in the system design or, if that is not possible, to prevent or control them in the system or component design. This process of scenario development is exactly the goal of hazard analysis and STPA. As an example of how the results of the analysis are used to refine the high-level safety constraints, consider the second high-level TCAS constraint: that TCAS must not provide resolution advisories that degrade vertical separation between two aircraft (cause an NMAC):

SC.7: *TCAS must not create near misses (result in a hazardous level of vertical separation that would not have occurred had the aircraft not carried TCAS) (→ H1).*

SC.7.1: *Crossing Maneuvers must be avoided if possible (↓ 2.36, ↓ 2.38, ↓ 2.48, ↓ 2.49.2).*

SC.7.2: *The reversal of a displayed advisory must be extremely rare*[4] *(↓ 2.51, ↓ 2.56.3, ↓ 2.65.3, ↓ 2.66).*

SC.7.3: *TCAS must not reverse an advisory if the pilot will have insufficient time to respond to the RA before the closest point of approach (four seconds*

4. This requirement is clearly vague and untestable. Unfortunately, I could find no definition of "extremely rare" in any of the TCAS documentation to which I had access.

or less) or if own and intruder aircraft are separated by less than 200 feet vertically when ten seconds or less remain to closest point of approach (↓ 2.52).

Note again that pointers are used to trace these constraints into the design features used to implement them.

10.3.8 System Design and Analysis

Once the basic requirements and design constraints have been at least partially specified, the system design features that will be used to implement them must be created. A strict top-down design process is, of course, not usually feasible. As design decisions are made and the system behavior becomes better understood, additions and changes will likely be made in the requirements and constraints. The specification of assumptions and the inclusion of traceability links will assist in this process and in ensuring that safety is not compromised by later decisions and changes. It is surprising how quickly the rationale behind the decisions that were made earlier is forgotten.

Once the system design features are determined, (1) an internal control structure for the system itself is constructed along with the interfaces between the components and (2) functional requirements and design constraints, derived from the system-level requirements and constraints, are allocated to the individual system components.

System Design

What has been presented so far in this chapter would appear in level 1 of an intent specification. The second level of an intent specification contains *System Design Principles*—the basic system design and scientific and engineering principles needed to achieve the behavior specified in the top level, as well as any derived requirements and design features not related to the level 1 requirements.

While traditional design processes can be used, STAMP and STPA provide the potential for safety-driven design. In safety-driven design, the refinement of the high-level hazard analysis is intertwined with the refinement of the system design to guide the development of the system design and system architecture. STPA can be used to generate safe design alternatives or applied to the design alternatives generated in some other way to continually evaluate safety as the design progresses and to assist in eliminating or controlling hazards in the emerging design, as described in chapter 9.

For TCAS, this level of the intent specification includes such general principles as the basic *tau* concept, which is related to all the high-level alerting goals and constraints:

2.2: *Each TCAS-equipped aircraft is surrounded by a protected volume of airspace. The boundaries of this volume are shaped by the tau and DMOD criteria (↑1.20.3).*

2.2.1: *TAU: In collision avoidance, time-to-go to the closest point of approach (CPA) is more important than distance-to-go to the CPA. Tau is an approximation of the time in seconds to CPA. Tau equals 3600 times the slant range in nmi, divided by the closing speed in knots.*

2.2.2: *DMOD: If the rate of closure is very low, a target could slip in very close without crossing the tau boundaries and triggering an advisory. In order to provide added protection against a possible maneuver or speed change by either aircraft, the tau boundaries are modified (called DMOD). DMOD varies depending on own aircraft's altitude regime (→ 2.2.4).*

The principles are linked to the related higher-level requirements, constraints, assumptions, limitations, and hazard analysis as well as to lower-level system design and documentation and to other information at the same level. Assumptions used in the formulation of the design principles should also be specified at this level.

For example, design principle 2.51 (related to safety constraint SC-7.2 shown in the previous section) describes how sense[5] reversals are handled:

2.51: *Sense Reversals: (↓ Reversal-Provides-More-Separation) In most encounter situations, the resolution advisory will be maintained for the duration of an encounter with a threat aircraft (↑SC-7.2). However, under certain circumstances, it may be necessary for that sense to be reversed. For example, a conflict between two TCAS-equipped aircraft will, with very high probability, result in selection of complementary advisory senses because of the coordination protocol between the two aircraft. However, if coordination communication between the two aircraft is disrupted at a critical time of sense selection, both aircraft may choose their advisories independently (↑HA-130). This could possibly result in selection of incompatible senses (↑HA-395).*

2.51.1: . . . *[information about how incompatibilities are handled]*

Design principle 2.51 describes the conditions under which reversals of TCAS advisories can result in incompatible senses and lead to the creation of a hazard by TCAS. The pointer labeled HA-395 points to the part of the hazard analysis analyzing that problem. The hazard analysis portion labeled HA-395 would have a complementary pointer to section 2.51. The design decisions made to handle such

5. The *sense* is the direction of the advisory, such as descend or climb.

incompatibilities are described in 2.51.1, but that part of the specification is omitted here. 2.51 also contains a hyperlink (↓Reversal-Provides-More-Separation) to the detailed functional level 3 logic (component black-box requirements specification) used to implement the design decision.

Information about the allocation of these design decisions to individual system components and the logic involved is located in level 3, which in turn has links to the implementation of the logic in lower levels. If a change has to be made to a system component (such as a change to a software module), it is possible to trace the function computed by that module upward in the intent specification levels to determine whether the module is safety critical and if (and how) the change might affect system safety.

As another example, the TCAS design has a built-in bias against generating advisories that would result in the aircraft crossing paths (called *altitude crossing advisories*).

> **2.36.2:** *A bias against altitude crossing RAs is also used in situations involving intruder level-offs at least 600 feet above or below the TCAS aircraft (↑SC.7.1). In such a situation, an altitude-crossing advisory is deferred if an intruder aircraft that is projected to cross own aircraft's altitude is more than 600 feet away vertically (↓ Alt_Separation_Test).*
>
> **Assumption:** *In most cases, the intruder will begin a level-off maneuver when it is more than 600 feet away and so should have a greatly reduced vertical rate by the time it is within 200 feet of its altitude clearance (thereby either not requiring an RA if it levels off more than* ZTHR[6] *feet away or requiring a non-crossing advisory for level-offs begun after* ZTHR *is crossed but before the 600 foot threshold is reached).*

Again, the example above includes a pointer down to the part of the black box component requirements (functional) specification (*Alt_Separation_Test)* that embodies the design principle. Links could also be provided to detailed mathematical analyses used to support and validate the design decisions.

As another example of using links to embed design rationale in the specification and of specifying limitations (defined later) and potential hazardous behavior that could not be controlled in the design, consider the following. TCAS II advisories may need to be inhibited because of an inadequate climb performance for the particular aircraft on which TCAS is installed. The collision avoidance maneuvers posted as advisories (called RAs or resolution advisories) by TCAS assume an aircraft's ability to safely achieve them. If it is likely they are beyond the capability

6. The vertical dimension, called ZTHR, used to determine whether advisories should be issued varies from 750 to 950 feet, depending on the TCAS aircraft's altitude.

of the aircraft, then TCAS must know beforehand so it can change its strategy and issue an alternative advisory. The performance characteristics are provided to TCAS through the aircraft interface (via what are called *aircraft discretes*). In some cases, no feasible solutions to the problem could be found. An example design principle related to this problem found at level 2 of the TCAS intent specification is:

> **2.39:** *Because of the limited number of inputs to TCAS for aircraft, performance inhibits, in some instances where inhibiting RAs would be appropriate it is not possible to do so (↑L6). In these cases, TCAS may command maneuvers that may significantly reduce stall margins or result in stall warning (↑SC9.1). Conditions where this may occur include . . . The aircraft flight manual or flight manual supplement should provide information concerning this aspect of TCAS so that flight crews may take appropriate action* (↓ [Pointers to pilot procedures on level 3 and Aircraft Flight Manual on level 6).

Finally, design principles may reflect tradeoffs between higher-level goals and constraints. As examples:

> **2.2.3:** *Tradeoffs must be made between necessary protection (↑1.18) and unnecessary advisories (↑SC.5, SC.6). This is accomplished by controlling the sensitivity level, which controls the tau, and therefore the dimensions of the protected airspace around each TCAS-equipped aircraft. The greater the sensitivity level, the more protection is provided but the higher is the incidence of unnecessary alerts. Sensitivity level is determined by . . .*

> **2.38:** *The need to inhibit* CLIMB *RAs because of inadequate aircraft climb performance will increase the likelihood of TCAS II (a) issuing crossing maneuvers, which in turn increases the possibility that an RA may be thwarted by the intruder maneuvering (↑SC7.1, HA-115), (b) causing an increase in* DESCEND *RAs at low altitude (↑SC8.1), and (c) providing no RAs if below the descend inhibit level (1200 feet above ground level on takeoff and 1000 feet above ground level on approach).*

Architectural Design, Functional Allocation, and Component Implementation (Level 3)

Once the general system design concepts are agreed upon, the next step usually involves developing the design architecture and allocating behavioral requirements and constraints to the subsystems and components. Once again, two-way tracing should exist between the component requirements and the system design principles and requirements. These links will be available to the subsystem developers to be used in their implementation and development activities and in verification (testing and reviews). Finally, during field testing and operations, the links and recorded assumptions and design rationale can be used in safety change analysis, incident and accident analysis, periodic audits, and performance monitoring as required to ensure that the operational system is and remains safe.

Level 3 of an intent specification contains the system architecture, that is, the allocation of functions to components and the designed communication paths among those components (including human operators). At this point, a black-box functional requirements specification language becomes useful, particularly a formal language that is executable. SpecTRM-RL is used as the example specification language in this section [85, 86]). An early version of the language was developed in 1990 to specify the requirements for TCAS II and has been refined and improved since that time. SpecTRM-RL is part of a larger specification management system called SpecTRM (Specification Tools and Requirements Methodology). Other languages, of course, can be used.

One of the first steps in low-level architectural design is to break the system into a set of components. For TCAS, only three components were used: surveillance, collision avoidance, and performance monitoring.

The environment description at level 3 includes the assumed behavior of the external components (such as the altimeters and transponders for TCAS), including perhaps failure behavior, upon which the correctness of the system design is predicated, along with a description of the interfaces between the TCAS system and its environment. Figure 10.11 shows part of a SpecTRM-RL description of an environment component, in this case an altimeter.

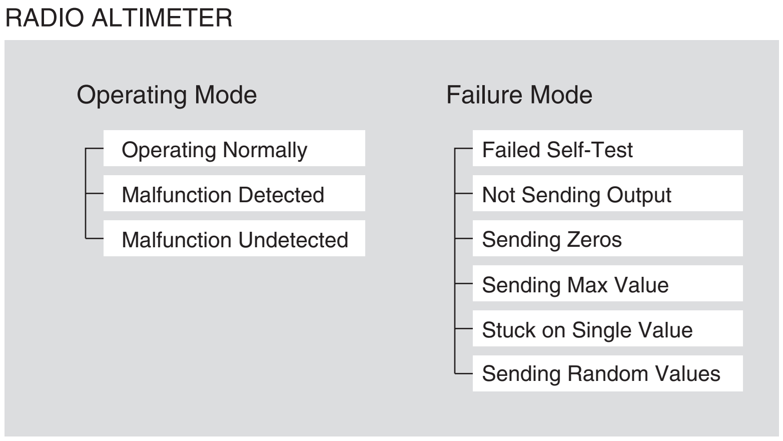
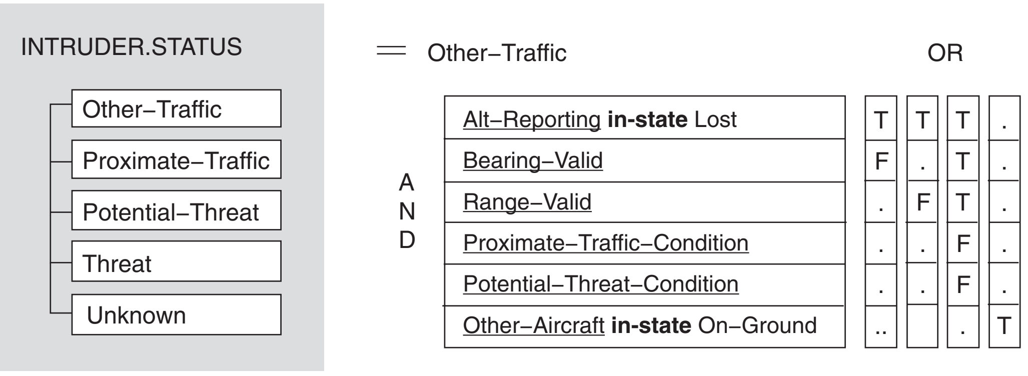


Figure 10.11
Part of the SpecTRM-RL description of an environment component (a radio altimeter). Modeling failure behavior is especially important for safety analyses. In this example, (1) the altimeter may be operating correctly, (2) it may have failed in a way that the failure can be detected by TCAS II (that is, it fails a self-test and sends a status message to TCAS or it is not sending any output at all), or (3) the malfunctioning is undetected and it sends an incorrect radio altitude.



<u>Alt–Reporting</u> **in-state** Lost	T	T	T	.
<u>Bearing–Valid</u>	F	.	T	.
<u>Range–Valid</u>	.	F	T	.
<u>Proximate–Traffic–Condition</u>	.	.	F	.
<u>Potential–Threat–Condition</u>	.	.	F	.
<u>Other–Aircraft</u> **in-state** On–Ground	..		.	T

Description: A threat is reclassified as other traffic if its altitude reporting has been lost (↑<u>2.13</u>) and either the bearing or range inputs are invalid; if its altitude reporting has been lost and both the range and bearing are valid but neither the proximate nor potential threat classification criteria are satisfied; or the aircraft is on the ground (↑<u>2.12</u>).

Mapping to Level 2: ↑<u>2.23</u>, ↑<u>2.29</u>

Mapping to Level 4: ↑<u>4.7.1</u>, Traffic–Advisory

Figure 10.12
Example from the level 3 SpecTRM-RL model of the collision avoidance logic. It defines the criteria for downgrading the status of an intruder (into our protected volume) from being labeled a threat to being considered simply as other traffic. Intruders can be classified in decreasing order of importance as a threat, a potential threat, proximate traffic, and other traffic. In the example, the criterion for taking the transition from state *Threat* to state *Other Traffic* is represented by an AND/OR table, which 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 a dot represent "don't care" conditions.

A system is an abstraction and the system boundaries can be set anywhere convenient for the purposes of the specifier. In this example, the environment includes any component that was already on the aircraft or in the airspace control system and was not newly designed or built as part of the TCAS effort.

All communications between the system and external components need to be described in detail, including the designed interfaces. The black-box behavior of each component also needs to be specified. This specification serves as the functional requirements for the components. What is included in the component specification will depend on whether the component is part of the environment or part of the system being constructed. Figure 10.12 shows part of the SpecTRM-RL description of the behavior of the CAS (collision avoidance system) subcomponent. SpecTRM-RL specifications are intended to be both easily readable with minimum instruction and formally analyzable. They are also executable and can be used in a system simulation environment. Readability was a primary goal in the design of SpecTRM-RL, as was completeness with regard to safety. Most of the requirements completeness criteria described in *Safeware* and rewritten as functional design principles in chapter 9 of this book are included in the syntax of the language to assist in system safety reviews of the requirements.

SpecTRM-RL explicitly shows the process model used by the controller and describes the required behavior in terms of this model. A state machine model is used to describe the system component's process model, in this case the state of the aircraft and the air space around it, and the ways the process model can change state.

Logical behavior is specified in SpecTRM-RL using AND/OR tables. Figure 10.12 shows a small part of the specification of the TCAS collision avoidance logic. For TCAS, an important state variable is the status of the other aircraft around the TCAS aircraft, called *intruders*. Intruders are classified into four groups: Other Traffic, Potential Threat, and Threat. The figure shows the logic for classifying an intruder as Other Traffic using an AND/OR table. The information in the tables can be visualized in additional ways.

The rows of the table represent AND relationships, while the columns represent OR. The state variable takes the specified value (in this case, *Other Traffic*) if any of the columns evaluate to TRUE. A column evaluates to TRUE if all the rows have the value specified for that row in the column. A dot in the table indicates that the value for the row is irrelevant. Underlined variables represent hyperlinks. For example, clicking on "<u>Alt Reporting</u>" would show how the Alt Reporting variable is defined: In our TCAS intent specification[7] [121], the altitude report for an aircraft is defined as *Lost* if no valid altitude report has been received in the past six seconds. Bearing Valid, Range Valid, Proximate Traffic Condition, and Proximate Threat Condition are *macros*, which simply means that they are defined using separate logic tables. The additional logic for the macros could have been inserted here, but sometimes the logic gets very complex and it is easier for specifiers and reviewers if, in those cases, the tables are broken up into smaller pieces (a form of refinement abstraction). This decision is, of course, up to the creator of the table.

The behavioral descriptions at this level are purely black-box: They describe the inputs and outputs of each component and their relationships *only* in terms of externally visible behavior. Essentially it represents the transfer function across the component. Any of these components (except the humans, of course) could be implemented either in hardware or software. Some of the TCAS surveillance

7. A SpecTRM-RL model of TCAS was created by the author and her students Jon Reese, Mats Heimdahl, and Holly Hildreth to assist in the certification of TCAS II. Later, as an experiment to show the feasibility of creating intent specifications, the author created the level 1 and level 2 intent specification for TCAS. Jon Reese rewrote the level 3 collision avoidance system logic from the early version of the language into SpecTRM-RL.

functions are, in fact, implemented using analog devices by some vendors and digital by others. Decisions about physical implementation, software design, internal variables, and so on are limited to levels of the specification below this one. Thus, this level serves as a rugged interface between the system designers and the component designers and implementers (including subcontractors).

Software need not be treated any differently than the other parts of the system. Most safety-related software problems stem from requirements flaws. The system requirements and system hazard analysis should be used to determine the behavioral safety constraints that must be enforced on software behavior and that the software must enforce on the controlled system. Once that is accomplished, those requirements and constraints are passed to the software developers (through the black-box requirements specifications), and they use them to generate and validate their designs just as the hardware developers do.

Other information at this level might include flight crew requirements such as description of tasks and operational procedures, interface requirements, and the testing requirements for the functionality described on this level. If the black-box requirements specification is executable, system testing can be performed early to validate requirements using system and environment simulators or hardware-in-the-loop simulation. Including a visual operator task-modeling language permits integrated simulation and analysis of the entire system, including human–computer interactions [15, 177].

Models at this level are reusable, and we have found that these models provide the best place to provide component reuse and build component libraries [119]. Reuse of application software at the code level has been problematic at best, contributing to a surprising number of accidents [116]. Level 3 black-box behavioral specifications provide a way to make the changes almost always necessary to reuse software in a format that is both reviewable and verifiable. In addition, the black-box models can be used to maintain the system and to specify and validate changes before they are made in the various manufacturers' products. Once the changed level 3 specifications have been validated, the links to the modules implementing the modeled behavior can be used to determine which modules need to be changed and how. Libraries of component models can also be developed and used in a plug-and-play fashion, making changes as required, in order to develop product families [211].

The rest of the development process, involving the implementation of the component requirements and constraints and documented at levels 4 and 5 of intent specifications, is straightforward and differs little from what is normally done today.

10.3.9 Documenting System Limitations

When the system is completed, the system limitations need to be identified and documented. Some of the identification will, of course, be done throughout the development. This information is used by management and stakeholders to determine whether the system is adequately safe to use, along with information about each of the identified hazards and how they were handled.

Limitations should be included in level 1 of the intent specification, because they properly belong in the customer view of the system and will affect both acceptance and certification.

Some limitations may be related to the basic functional requirements, such as these:

L4: *TCAS does not currently indicate horizontal escape maneuvers and therefore does not (and is not intended to) increase horizontal separation.*

Limitations may also relate to environment assumptions. For example:

L1: *TCAS provides no protection against aircraft without transponders or with nonoperational transponders (→EA3, HA-430).*

L6: *Aircraft, performance limitations constrain the magnitude of the escape maneuver that the flight crew can safely execute in response to a resolution advisory. It is possible for these limitations to preclude a successful resolution of the conflict (→H3, ↓2.38, 2.39).*

L4: *TCAS is dependent on the accuracy of the threat aircraft's reported altitude. Separation assurance may be degraded by errors in intruder pressure altitude as reported by the transponder of the intruder aircraft (→EA5).*

Assumption: *This limitation holds for the airspace existing at the time of the initial TCAS deployment, where many aircraft use pressure altimeters rather than GPS. As more aircraft install GPS systems with greater accuracy than current pressure altimeters, this limitation will be reduced or eliminated.*

Limitations are often associated with hazards or hazard causal factors that could not be completely eliminated or controlled in the design. Thus they represent accepted risks. For example,

L3: *TCAS will not issue an advisory if it is turned on or enabled to issue resolution advisories in the middle of a conflict (→ HA-405).*

L5: *If only one of two aircraft is TCAS equipped while the other has only ATCRBS altitude-reporting capability, the assurance of safe separation may be reduced (→ HA-290).*

In the specification, both of these system limitations would have pointers to the relevant parts of the hazard analysis along with an explanation of why they could not be eliminated or adequately controlled in the system design. Decisions about deployment and certification of the system will need to be based partially on these limitations and their impact on the safety analysis and safety assumptions of the encompassing system, which, in the case of TCAS, is the overall air traffic system.

A final type of limitation is related to problems encountered or tradeoffs made during system design. For example, TCAS has a high-level performance-monitoring requirement that led to the inclusion of a self-test function in the system design to determine whether TCAS is operating correctly. The following system limitation relates to this self-test facility:

L9: *Use by the pilot of the self-test function in flight will inhibit TCAS operation for up to 20 seconds depending upon the number of targets being tracked. The ATC transponder will not function during some portion of the self-test sequence (↓6.52).*

These limitations should be linked to the relevant parts of the development and, most important, operational specifications. For example, L9 may be linked to the pilot operations manual.

10.3.10 System Certification, Maintenance, and Evolution

At this point in development, the safety requirements and constraints are documented and traced to the design features used to implement them. A hazard log contains the hazard information (or links to it) generated during the development process and the results of the hazard analysis performed. The log will contain embedded links to the resolution of each hazard, such as functional requirements, design constraints, system design features, operational procedures, and system limitations. The information documented should be easy to collect into a form that can be used for the final safety assessment and certification of the system.

Whenever changes are made in safety-critical systems or software (during development or during maintenance and evolution), the safety of the change needs to be reevaluated. This process can be difficult and expensive if it has to start from scratch each time. By providing links throughout the specification, it should be easy to assess whether a particular design decision or piece of code was based on the original safety analysis or safety-related design constraint and only that part of the safety analysis process repeated or reevaluated.