Using Fault Modeling in Safety Cases

Robyn Lutz
Jet Propulsion Lab/Caltech and Iowa State University
rlutz@cs.iastate.edu

Ann Patterson-Hine
Discovery and Systems Health
NASA Ames Research Center
Ann.Patterson-Hine@nasa.gov

Abstract

For many safety-critical systems a safety case is built as part of the certification or acceptance process. The safety case assembles evidence to justify that the design and implementation of a system avoid hazardous software behavior. Fault modeling and analysis can provide a rich source of evidence that the design meets safety goals. However, there is currently little guidance available to bridge the gap between the fault modeling that developers perform and the mandated safety case. In this experience report we describe results and open issues from an investigation of how evidence from software tool-supported fault modeling and analysis of a spacecraft power system could assist in safety-case construction. The ways in which the software fault models can provide evidence for the safety case appears to be applicable to other critical systems.

1. Introduction

Preventing hazardous software behavior is a major concern to developers of safety-critical systems. Safety-critical software is software that can cause, or that can prevent, a hazardous situation or loss of mission from occurring. Examples of safety-critical software include the software in implantable medical devices, in smart vehicles, in assistive robots, and in spacecraft.

For many safety-critical systems a safety case must be built as part of the certification process [1]. The safety case provides evidence to justify that the design and implementation of the system avoid hazardous software behavior in its intended environment. It links system information, called evidence, with the safety requirements, called safety claims, via arguments which show the relationship between the requirements and the system information.

A fault model represents how the system detects and handles component and communication failures that could contribute to unsafe situations. Fault models, together with the automated analysis and simulation that they enable, document the ways in which a system can fail and how the design detects and recovers from those failures [5]. This information is a rich and largely untapped source of evidence that the design meets its safety goals prior to implementation and deployment. The models support automated analysis of monitorability (that certain undesirable states can be detected and uniquely identified) and of controllability (that the system can autonomously return to a safe state).

While it is clear that information in the fault models can assist in determining whether the safety requirements on the system are satisfied, there is little guidance available as to *how to incorporate information from the fault models into the safety case.* The motivation for this work was to investigate whether the data provided by the fault modeling and analysis during design development of a spacecraft power system could ease construction of a safety case. In this way the safety case could begin to be assembled in parallel with the development of the system, perhaps both enhancing the arguments and reducing the effort required to make the safety case.

The contribution of this paper is to describe the ways in which design-phase fault models can be used to help assemble a safety case and to report preliminary findings and difficulties encountered. We structure this summary of our experience around three questions:

1. How readily can the fault models be used as *evidence* for a safety case? This question addresses the applicability of the models as evidence.
2. How can we use or adjust the modeling process to ease the construction of safety-case *arguments*? This question links the artifacts (models, model analysis results, and model

1071-9458/08 $25.00 © 2008 IEEE
DOI 10.1109/ISSRE.2008.13

simulation) to satisfaction of the safety requirements.

3. How well do steps to support construction of the safety case *align* with the developer's interests? This question investigates whether a case can be made to the modelers (who are primarily interested in achieving a good design) that measures to ease documentation of a safety case also benefit them.

The application that we describe in this paper is the Advanced Diagnostics and Prognostics Testbed (ADAPT), an electrical power system testbed under development at NASA Ames [2]. Its data acquisition and control system sends commands to and receives data from the electrical power system. ADAPT currently supports sixteen power loads. On an inhabited spacecraft some of these loads will be safety-critical, such as air pumps and lights. Our description of a safety case for ADAPT is exploratory and only for investigative purposes. ADAPT is not safety critical and does not require a safety case, but provides an evaluation platform for NASA systems that will be safety critical and may require them.

Preliminary results reported here indicate that the ADAPT fault model can provide the following types of evidence for a safety case:

- *Evidence from the contingency analysis*, used in constructing the fault model. These results provide information about the thoroughness of the analysis and form the baseline for arguments that all envisioned contingencies are handled in the design.
- *Evidence from review of the fault model*. These results provide evidence that safety requirements to diagnose and recover from failures are satisfied in the design.
- *Evidence from tool-supported static analysis of the fault model.* In the application described here the modeling and analysis tool (TEAMS, described in Sect. 3), provided information about backward and forward fault propagation, and about undetectable and indistinguishable failures.
- *Evidence from running the fault model.* In the application described here, we did not produce executable specifications, but another version of the tool, called TEAMS-RT, can run dynamic checks on the TEAMS model. In previous work with a UAV we also used executable specifications [8].

Ideally, a safety case is not an ex post facto document but a chain of reasoning that is assembled in parallel with the software development. In the rest of the paper we describe how the fault models, together with the analyses used to build them and to analyze them, can be usefully and readily incorporated into the safety case to provide assurance about the system's design. Design information is clearly only one small piece of a safety case, but it is an essential one.

The remainder of the paper is organized as follows. Section 2 briefly describes safety cases and related work. Section 3 reports how the fault-modeling and analysis results for ADAPT can be justifiably used as evidence for a safety case. Section 4 summarizes our experience and some open issues in the context of the three questions posed above (usefulness as evidence, construction of safety-case arguments, and alignment with developers' interests). Section 5 provides a conclusion.

2. Safety cases

For some systems, construction of a safety case is required, often as part of an application for certification by a government agency. However, interest in safety cases extends beyond the systems for which they are legally mandated. The planned return to the Moon has focused attention at NASA on safety and dependability cases as candidate techniques for enhancing the safety of manned missions and the dependability of those systems.

A safety case typically consists of three main parts: claims, evidence, and arguments that link the claims to the evidence [1, 6.]. Claims are the safety requirements that the system must satisfy. The evidence is information that helps demonstrate that the safety requirements have been met. The arguments show how the evidence relates to the claims. The chain of reasoning justifies why we can place trust in the system operating safely in its planned environment [6]. Habli, Wu, Attwood and Kelly have recently extended the notion of argument to the requirements phase [4]. We here use Kelly's Goal Structuring Notation (GSN) [6] to graphically represent and relate the claims to the evidence.

3. Fault modeling as evidence

We studied the spacecraft power system testbed as part of an investigation into contingency analysis of autonomous software systems. Such systems handle not only faults but also other unexpected environmental or operational scenarios that might add operational risk. These broader classes of anomalies that must be anticipated and handled are called contingencies [8]. Contingencies that impede safety requirements are hazards. An example of a safety-related contingency is "failure to provide power to a critical load".

The approach to modeling was four-fold: (1) identify the contingencies, (2) identify the requirements to detect and handle them, (3) represent the contingencies of concern in an integrated system/fault design model, and (4) analyze the model to identify contingencies that are not adequately diagnosed or handled. Application of each of these steps to ADAPT is described below. In previous work we have applied this approach to the vision subsystem of two unpiloted aerial vehicles (UAVs) [8] and to the critical pointing software for Mars spacecraft [7].

Step 1: Contingency Analysis. The first step of the contingency-analysis process is to identify the contingencies. We used bi-directional safety analysis which combines forward analysis using software failure modes, effects & criticality analysis (S-FMECA) and backward analysis, using software fault tree analysis (S-FTA). The S-FMECAs and S-FTAs produced identify hazards against which the system must protect. The ADAPT project also developed failure scenarios, such as losing one battery or having a load sensor fail, that described situations the system was required to handle.

Step 2: Safety Requirements. To identify the software safety requirements for monitorability and controllability from the contingencies, we used goal-oriented analysis [13]. This structured the investigation and evaluation of alternative resolutions to the contingencies and provided a way to document the decisions. For example, for the contingency of a power storage device failure, a safety requirement is to power up a redundant load on another branch of the power system. Traceability from the contingencies to the safety requirements is explicit, so the safety-case chain of reasoning is verifiable.

Step 3: Fault modeling. A TEAMS fault model consists of hierarchical components and connectors. To create the fault model, the architectural components (subsystems and their modules) associated with the key functionalities and their potential failures were represented, together with their properties (e.g., input and output). The output from Steps 1 and 2 inform the fault modeling.

Each failure mode of a component is represented as a labeled box at the lowest level of the model (e.g., "load relay fails open"). Boolean-valued checks (called tests) are placed at appropriate points to monitor for the occurrence of the failures. On ADAPT these checks are currently implemented in hardware or performed by manual inspection, but will be largely handled autonomously in software in future builds [3]. The connectors along which data signals propagate trace the module dependencies.

We used the TEAMS modeling tool from QSI, Inc. both because it was developed to support the modeling and automated troubleshooting of critical systems and because it is used at several NASA centers. Modelers at QSI subsequently used the TEAMS toolset to build a large fault model of ADAPT. The model contained 737 failure sources in 307 components with 275 modeled diagnostic tests.

Step 4: Automated Analysis of the Fault Model. While review of the model provides evidence of traceability among the steps, the automated analyses provide evidence of completeness (or gaps) in coverage of the contingencies. To verify the ADAPT design for diagnosis and handling of the contingencies we used several tool-supported capabilities in TEAMS. One of the most useful is the auto-generated diagnostic tree. TEAMS uses a dependency matrix to produce these trouble-shooting diagnostic trees. A diagnostic tree describes an optimized sequence of checks based on the results of prior checks. Different operational modes or configurations have different diagnostic trees since some faults only occur in some modes, and the available signals and checks depend on the operational mode and configuration.

The trees provided *evidence of monitorability* by showing whether each modeled failure could be uniquely identified with available sensor data and checks. The diagnostic trees gave *evidence of completeness* by listing failures that were undetectable with available sensor data and checks.

4. Experience and open issues

At the beginning of the paper we posed three questions, which we repeat here in abbreviated form:

- How readily can the fault modeling and analysis results be used as *evidence* for a safety case?
- How can we adjust the modeling process to ease the *construction* of safety-case arguments?
- How well do efforts to support the safety case *align* with the developer's interests?

In this section we discuss our experience to date and the open issues encountered for each question.

4.1 Using fault-modeling results as evidence

Figure 1 shows a portion of a safety case for the ADAPT electrical power system. The ADAPT development process that we describe here produced the four types of evidence described in Sect. 1 and shown in the leaf nodes of Fig. 1. The top-level

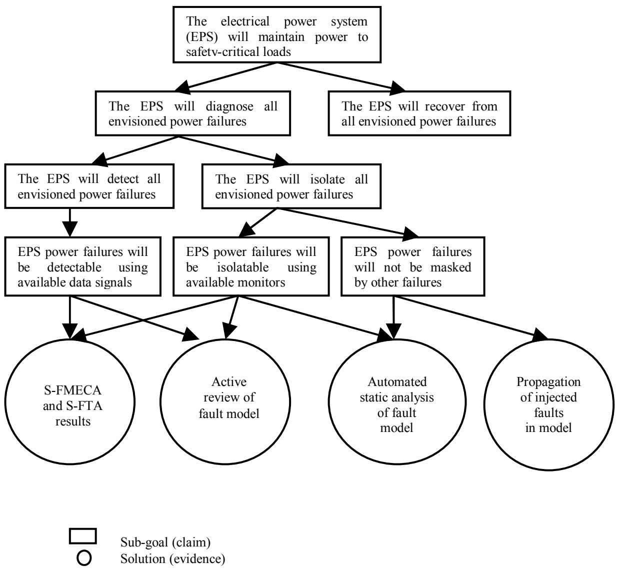
The electrical power system (EPS) will maintain power to safety-critical loads
The EPS will diagnose all envisioned power failures
The EPS will recover from all envisioned power failures
The EPS will detect all envisioned power failures
The EPS will isolate all envisioned power failures
EPS power failures will be detectable using available data signals
EPS power failures will be isolatable using available monitors
EPS power failures will not be masked by other failures
S-FMECA and S-FTA results
Active review of fault model
Automated static analysis of fault model
Propagation of injected faults in model
Sub-goal (claim)
Solution (evidence)

Figure 1. Portion of goal structure for ADAPT

safety claim is that "The electrical power system (EPS) will maintain power to safety-critical loads." This is refined into two subgoals, the leftmost of which is then successively refined in the figure.

The arrows in Fig. 1 trace the line of reasoning from the top-level safety claim to the evidence. The four leaf nodes show the available evidence that justifies belief that the associated subgoals are met. For example, the evidence described in the first leaf node is from the S-FMECA tables and fault trees produced in the contingency analysis (i.e., Step 1). We focus here on the evidence in the third leaf node, i.e., the auto-generated TEAMS reports from analysis of the fault model (Step 4). One important piece of context information (not shown in the figure but needed for the argument) is that some faults are only possible, and some checks are only appropriate or available, in certain configurations or modes of operation.

To justify use of the fault model and analysis results as evidence, we must demonstrate that the fault model accurately represents the as-built system. The ADAPT fault model was verified against the actual ADAPT system through review of the model by domain experts on the project, through simulation of normal and failure scenarios in the model with comparison of results to actual ADAPT outputs, and through extensive exercising of the model. This is shown in the second leaf node in Fig. 1.

For ADAPT the fault models and analysis results gave useful design-level evidence for a safety case. TEAMS-generated diagnostic results included identification of (1) failures that were undetectable with the available checks (i.e., had a row with no entries in the diagnostic matrix, (2) failures that were indistinguishable with the available checks (called an "ambiguity group" in TEAMS, and (3) failures that had unintended redundant checks (i.e., had two or more identical rows in the diagnostic matrix). We give an example of each below.

An example of an *undetectable failure* was that the health of an AC Load could not be known with current checks. The solution, to add an additional sensor, yielded evidence that failure of this critical load could now be detected in the design. An example of an *indistinguishable failure* was that there was no way to isolate between the failure of a load relay controller fault and a voltage sensor fault. This was of interest because it forced a coarser-grained recovery (switching to a backup load) where a simpler recovery might have sufficed. An example of *unintended redundancy in the checks* was that two monitors (one checked the health of a light sensor and one of a temperature sensor) inadvertently both indicated a failure when any one of the same three upstream components failed. Instead, the intent was for each to contribute unique diagnostic information.

To summarize, the auto-generated information regarding potential failures, detection mechanisms, recovery strategies, fault propagation, fault coverage metrics, and undetectable faults was useful and relevant to the safety case. The information served as evidence that the safety requirements were met in the design. This usage also seems to be consistent with the Adelard Safety Case Development Manual's advice that a preliminary safety case argument to show that the candidate design satisfies the safety-related arguments can use evidence from the system hazard analysis (e.g., FTAs) and from qualitative design assessment studies (e.g., of analyzability) [1].

4.2 Constructing safety-case arguments

An advantage of making the relationships between the diagnostic model and the safety case explicit is that this can be done to some extent in parallel with development. In this way the fault model is used for two purposes: (1) to verify and improve the design of the software's contingency detection and remediation, and (2) to document for the safety case evidence that safety goals are met. It is the traceability from the goal to the contingency to the resolution to the design that provides the structure of the argument. In the terminology of the safety case, this traceability is a safety-case strategy.

The TEAMS model was locally very readable and, due to the hierarchical representation, easy to understand at any level. One challenge we faced was that, as ADAPT grew, the model and number of failure scenarios also grew, making review of the model harder. We sometimes had difficulty not getting lost in the details. A related challenge was that some of our inquiries required drawing together an understanding of different parts of the model at different levels of detail. That is, the level of granularity needed to discuss an issue was often not uniform across the model. In such cases the information hiding provided in the model by the hierarchical nature of the representation sometimes worked against understanding.

Weinstock, Goodenough, and Hudak have similarly noted what they call "bulkiness" as a problem for dependability cases [10]. (Dependability cases are similar to safety cases but also consider tradeoffs among goals such as safety, availability, reliability, and security.) Joshi and Heimdahl also have described how large models are prone to "cluttering", which can make updating the model more error-prone [5]. Their suggestion is to specify the fault model separately from the model of the nominal behavior to reduce the information in each model. The risk is that separate development can sometimes lead to delays in considering faults and to inconsistencies among the models that are not caught until they are integrated.

4.2.1 Model slices. One approach that we tried and that improved our understanding of the design was the creation of a simplified model containing only the failure scenarios of concern and only the architectural elements (components, signals, testpoints) associated with them. For example, whereas the full ADAPT model had 737 failure sources in 307 components, and 275 tests (embedded monitors), one of the sliced models had just 33 failure sources in 33 components and 18 tests. Some contingencies of concern were that one load controller failed, that one voltage sensor failed, and that both the controller and sensor failed.

The sliced model also abstracted out some details that were not relevant to the concern being addressed. Failures that would have the same effect or response were grouped together. For example, the three failure modes modeled in the full model for Battery 1 (voltage low, impedance low, and over-temperature) were reduced in the sliced model to a single failure state. Redundant sensors were also removed where the duplication had no bearing on the current concern.

Targeted slicing (where the information to be hidden is selected based on its irrelevance to a specific contingency) may offer an approach to managing such "bulkiness" in safety cases but requires further investigation. For example, one sliced model focused on the failure of a voltage sensor in a single operating mode. This slicing was done manually by the QSI modelers, but we hope to be able to create such derivative models automatically in the future.

By creating a slice of the model we were able to remove the "clutter" while retaining a coherent view of how the nominal behavior is affected by the failure. For larger projects, it may also be the case that the slicing occurs naturally with incremental build-up of the models. *The sliced model helped focus attention on evidence that the design correctly diagnosed critical failures.*

4.2.2 Active reviews. Parnas and Weiss coined the term "active review" in recommending that design reviews be divided into multiple, small-group discussions that focus on one concern at a time, with each meeting using a set of questions posed by the designers, that the reviewers discuss with the designers [9].

There were several factors that encouraged the use of active reviews of the fault modeling on the ADAPT project. The model evolved rapidly as the testbed itself grew. Hardware components were added, removed, and replaced; instrumentation changed; and software simulation and fault injection functionalities were added. The people working on the different aspects of ADAPT were geographically distributed, so especially attentive to the danger of getting out of synch with each other. In addition, the TEAMS modeling and analysis tool we were using went through several updates as capabilities were added to it.

A series of small telephone conferences brought together the designers of the model, domain experts from the project, and the authors. On ADAPT the reviews were driven by questions drawn up by any of the parties. The ensuing discussions sometimes revealed important differences of opinion that affected the validity of the model or the appropriateness of the design. *Active design reviews helped us construct sound arguments.*

4.2.3 Multiple, consistent views. TEAMS stores, presents and exports analysis results both in forms that are human readable (e.g., directed acyclic graphs, Excel spreadsheets) and machine readable (e.g., XML). Consistency among the views is enforced automatically. This helps make the analytical results easy to understand and use as evidence.

In particular, we found it useful to be able to view the diagnostic design data both graphically and in a spreadsheet format. The effect was to have more ways of reviewing the model since different people studied the model using different views. *The multiple perspectives made the review process more thorough and, consequently, the evidence more trustworthy.*

4.3 Aligning safety-case effort with developers' interests

The project's interest in the fault modeling was for design verification rather than to support the construction of a safety case. It is thus appropriate to ask whether there are advantages for developers in beginning to build the safety case so early.

First, although we have assumed here that designers will not be involved in the construction of the safety case, this may be a false assumption, in which case any effort spent early to save time later can benefit them directly. Second, the additional overhead that we described above (documentation, multiple models, and active reviews) also contributes in a demonstrable way to the readability, maintainability, and quality of the fault modeling. The measures suggested above to assist in incrementally building the safety case (e.g., enhancing readability of models, storing design-phase fault-modeling analysis results, holding reviews focused on critical contingencies) also assist in verifying the completeness of the design mechanisms to detect and respond to failures. Thus, the complexity-management techniques that we used with the ADAPT models are not specific to this application and are also available when assembling evidence from fault models for safety cases in other systems.

More generally, we can look to Fig. 1 to gauge the extent to which the results presented here for deriving safety-case evidence from fault models are applicable to other projects. As can be seen by the leaf nodes in Fig. 1, automated static analysis results are only one source of evidence for the safety case. Thus, the results do not depend on the use of TEAMS. For example, the S-FMECA and S-FTA results in the leftmost leaf node, although more limited in their scope, are traditional sources of evidence. The advantage of the experience reported here is that it shows how to exploit fault modeling analysis results (preferably of a machine-checkable fault model) to enable an earlier start with less overhead to assembling the preliminary safety case.

5. Conclusion

The experience reported here shows how fault modeling and analysis results on a project provided design-phase evidence that could also ease construction of a safety case. The use of contingency analysis and tool-supported fault modeling in the design phase resulted in a more methodical approach to ensuring that designs met their safety claims. This, in turn, produced fault analysis results that were also appropriate for use in a safety case at an early stage without incurring additional overhead.

Acknowledgments. The authors thank Scott Poll, ADAPT Project Manager, and the researchers and developers at QSI. The research described in this paper was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, and Ames Research Center under a contract with the National Aeronautic and Space Administration and funded by NASA's OSMA Software Assurance Research Program. The first author's research is supported in part by NSF grants 0541163 and 0702758.

6. References

[1] *Adelard Safety Case Development Manual*, 2008.
[2] *Advanced Diagnostics and Prognostics Testbed (ADAPT) System Description, Operations, and Safety Manual*, NASA Ames Research Center, 2006.
[3] S. Ghosal, M. Azam, and V. Malepati, "FDIR on the ADAPTS Test Bed, Phase III Progress Report 1", Qualtech Systems, Inc, 2006.
[4] I. Habli, W. Wu, K. Attwood, and T. Kelly, "Extending argumentation to goal-oriented requirements engineering", *Advances in Conceptual Modeling – Foundations and Applications*, LNCS 4802, pp. 306-316.
[5] A. Joshi and M. P. E. Heimdahl, "Behavioral fault modeling for model-based safety analysis", HASE 2007, pp. 199-208.
[6] T. P. Kelly, and R. A. Weaver, "The goal structuring notation-a safety argument notation", *DSN Workshop on Assurance Cases*, 2004.
[7] R. Lutz, A. Patterson-Hine and A. Bajwa, "Tool-Supported Verification of Contingency Software Design in Evolving, Autonomous Systems", ISSRE'06, pp. 213-220.
[8] R. Lutz, A. Patterson-Hine, S. Nelson, C. Frost, D. Tal, and R. Harris, "Using obstacle analysis to identify contingency requirements on an unpiloted aerial vehicle", *REJ* 12(1), Jan. 2007, 41-54.
[9] D. L. Parnas and D.M. Weiss, "Active design reviews: principles and practices", *ICSE,* 1985, pp.132-136.
[10] C. B. Weinstock, J. B. Goodenough, and J. J. Hudak, "Dependability Cases", CMU/SEI-2004-TN-016, 2004.