Reasoning about Safety during Software Architecture Design

Tacksoo Im, Soujanya Vullam, and John D. McGregor
School of Computing
Clemson University
Clemson, SC USA
Email: {tim,svullam,johnmc}@cs.clemson.edu

Abstract
Architects use a variety of techniques to evaluate designs to determine the degree to which a product produced from the architecture would possess the desired levels of specific quality attributes. Reasoning frameworks are used to guide architecture definition by predicting the extent to which a software architecture satisfies its quality requirements. There has been much research about such direct runtime attributes as performance and modifiability but much less work has been done concerning such indirect attributes as safety.

We present a framework for reasoning about safety that is based on the observation that safety hazards sometimes lead to accidents when certain quality requirements of the system are not satisfied. This naturally leads to the use of reasoning frameworks for these other qualities as a means to indirectly reason about safety. We present our technique that utilizes standard safety engineering activities and a risk-based qualitative reasoning approach to make a judgment on the satisfaction of safety requirements by the architecture.

1 Introduction
Safety is an essential attribute of many dependable systems. Safety is defined as the freedom from unacceptable risk of an event that results in loss of life, injury, or property damage [11]. What is unacceptable varies from one domain to another. Software cannot cause direct harm but software embedded in automobiles, aircraft, robots, and even elevators has the potential for causing accidents, which result in injury or death. These accidents are the result of hazards, events that can potentially cause death or injury [11].

Many safety issues arise from the context in which the software is used but some issues arise due to a failure to achieve sufficient levels of certain quality attributes. For example, a failure to meet a performance goal may cause a synchronization problem, which results in loss of control at a critical moment. Even qualities concerning the static structure of the architecture, such as maintainability, can affect the safety of the system by reducing the rate at which defects are removed or increasing the rate at which defects are injected during maintenance.

Architects design an architecture to meet safety requirements by introducing certain structural elements and relationships. Certain architectural designs reduce the likelihood of a hazardous event from occurring and mitigate the impact of those accidents if they do occur [14]. The functionality and the external factors of the software are known to cause most safety problems. In order to address those problems more effectively, safety has to be considered starting with the architecture level.

At the earliest stages of safety analysis, system requirements, and the context in which the system will be deployed, are analyzed to identify hazards [12]. Certainly some accidents result from hardware failures over which the software has no control, but many accidents occur because of software failures. A safety analysis of the software examines the functional and non-functional software requirements. These requirements are analyzed to associate individual requirements with specific hazards. Our work on safety design is based on the idea that some safety requirements are closely related to other quality requirements.

The architect needs a method by which a proposed modification of the architecture can be compared to the original architecture to determine which will result in safer systems. This can be done by examining the effect of the modification on the safety hazards. To replace the older architecture, the modified architecture should either reduce the risk of certain events from occurring or it should result in those accidents that do occur having less impact. In this paper we present a technique for reasoning about safety during architecture definition that specifies the required analyses.

The main contribution of this paper is a descrip-
tion of a qualitative reasoning framework for safety. In particular we present a novel analytic theory for estimating the safety of systems built from the architecture. The theory relates the safety of a system to the system's ability to satisfy certain quality requirements. The theory uses traditional safety analysis techniques as a starting point and traces back to the causes of potential safety hazards.

In the next section we will provide some background on safety analysis and reasoning frameworks including our efforts with respect to reliability and confidentiality. We then provide a description of the safety reasoning framework. We apply the framework to an on-going case study. We conclude by describing the integration of safety analysis into a larger dependability reasoning framework and comparing our work to that of others.

2 Background

In this section we present a brief introduction to safety hazards and reasoning frameworks.

2.1 Safety Hazards

Safety hazards are identified using techniques such as Functional Hazard Analysis [16] [3]. This technique evaluates a representation of a system and its operational procedures to determine if humans or the environment will be exposed to hazards and take necessary measures to prevent loss [13].

A hazard has an associated risk, the possibility of suffering a loss [15]. From an architectural perspective, most, if not all, of the analysis revolves around that risk since the hazard is typically physically outside the software. For example, the anti-lock braking system in an automobile is a hazard since there is a risk the brakes fail to engage. The risk related to the braking software is managed in the software architecture.

Managing risks includes two types of activities: risk evaluation and risk mitigation. Risk evaluation of safety-related events (hazards) involves evaluating the severity of the loss that would result, the type of loss, and probability of such a loss occurring. These issues are considered while defining the safety requirements [2]. Thus, risk evaluation in the context of safety is instrumental in determining the safety requirements.

Risk mitigation comprises decreasing the probability of hazard occurrence and decreasing the intensity of loss caused by hazards. Our technique addresses the probability of occurrence but does not reduce the intensity of the loss. Modifying the architecture to increase the level of a quality reduces the probability of occurrence of certain hazards. For example, improving usability or security features in the architecture helps prevent safety problems such as users becoming distracted.

2.2 Reasoning frameworks

A reasoning framework is used by an architect to evaluate the value of a quality attribute in an architecture. “A reasoning framework provides the capability to reason about quality attribute behavior based on analytic theory [1].” A reasoning framework consists of six elements: problem description, analytic theory, model representation, interpretation, and evaluation procedure.

A reasoning framework takes a representation of the architecture as input and gives an estimate of the value of a quality attribute for a system built from the architecture as output. In a reasoning framework, quality scenarios describe how a use of the system affects the specific quality attribute. Scenarios are defined as brief narratives of expected use of a system from both development and end-user viewpoints [9]. The scenarios drive the identification of responsibilities which are one technique for architecture representation [1]. The responsibilities are indicative of the functions of the system, it can be an action, knowledge to be maintained, a decision to be carried out by a software system, or an element of that system [17].

3 Safety Reasoning Framework

Accidents are caused by failures (hazards that have been realized) [11] and there is an inherent nature to each failure [18], which in some cases can be related to qualities of the software involved in the failure. A failure of omission can indicate an availability problem, a failure to produce a correct value can indicate a reliability problem, and timing issues are indicative of performance problems [18]. Even failures that result from usability problems can lead to safety problems [10].

The “safety” quality will be defined in terms of the qualities identified as potential safety hazards during the analyses. We hypothesize that safety can be analyzed using a set of reasoning frameworks depending on the nature of the hazards and then the aggregate result from the reasoning frameworks can be used to characterize the level of safety in the architecture. This is because the failures that cause many safety problems can each be classified as a failure to satisfy a specific quality requirement. Any quality attribute that can increase the probability of a hazard resulting in an accident should be part of the safety analysis. For example, the safety of a software-based fire alarm
system is largely dependent on how reliably the fire is detected and processed by the system. This is an example where a failure to satisfy a reliability requirement becomes a safety problem.

In this section, we present a safety reasoning framework which leverages this observation while codifying some of the existing techniques and approaches. We first outline the actions in a traditional safety analysis and then we will use the canonical outline for describing reasoning frameworks to describe our framework. These work together to provide the safety analysis process shown in Figure 1.

3.1 Initial Safety Analyses

Our safety reasoning framework builds on the information collected in a traditional system safety analysis which includes Functional Hazard Analysis and Fault Tree Analysis. These analyses are subjective and directly affected by the expertise of the personnel conducting the analyses.

Functional Hazard Analysis (FHA) identifies hazards by examining the potential effects of functional failures [16]. Those that can result in injury or death are safety hazards and are the input to the Fault Tree Analysis. The potential effect of a functional failure will also help determine the root cause of the failure. The information from the FHA is captured in the form shown in Table 4 which shows the functional hazard analysis for our example system.

Fault Tree Analysis (FTA) identifies the root causes of undesired events including those events that result in injury or death. For our purposes only those hazards that are identified as safety critical during the Functional Hazard Analysis are subjected to FTA. FTA uses the graphical symbology shown in Figure 3 that identifies different types of events and linkages.

The Fault Trees for the safety critical hazards related to a specific product are an input into our reasoning framework. In our reasoning framework we will be looking for basic events, those circles at the bottom of the diagram in Figure 3.

3.2 Problem Description

Accidents occur as a result of failures during system execution, but may be caused by either dynamic or static faults in the architecture. Table 1 illustrates some of the failures that can affect the probability of a hazard resulting in an accident. This is not a comprehensive list. Almost any quality attribute can, in the right circumstances, affect safety. The architect is responsible for judging which quality attributes are a safety concern for the system that is under consideration. Although this may seem subjective, there are examples, such as, the Architecture Trade-off Analysis Method (ATAM), which show a similar approach to identifying qualities.

The input to the safety reasoning framework is the set of safety hazards output from the initial safety analysis described in section 3.1. A safety scenario is constructed for each of the identified safety hazards. A system safety requirement sets an expected tolerance for each hazard and the requirement is captured in a safety scenario.

The general scenario for the safety reasoning framework, which defines the form of a safety scenario, contains the information described in Table 2.

3.3 Analytic Theory

The analytic theory for the safety reasoning framework involves a set of transformations among the analytic theories related to the identified qualities. As stated previously, words in the description of a safety scenario, such as fault, missed deadline, etc, are used by the architect to map into quality attributes such as availability, reliability, or some other quality. The safety scenario is transformed into sce-
A stimulus that addresses a quality requirement that can increase the likelihood of an accident from a hazard.

Any actor of the system

The stimulus is introduced during the execution of the system.

System and potentially some real-world accident that results in injury or death

The system fails and the user is harmed in some way.

What is the impact of the failure?

### Table 2: General Scenario for Safety

As shown in Figure 2. This transformation from a safety scenario to a scenario for another quality attribute is done by semantically matching keywords in the scenario. For example, if a word related to scheduling is found in the safety scenario, then it is matched to a performance scenario. This transformation allows the safety reasoning framework to utilize the results of the target reasoning frameworks.

The transformation is further defined by identifying the information required for the general scenario of each target reasoning framework. The architect acquires any information needed for the quality scenario that is not already available in the safety framework. For example, the performance reasoning framework needs a maximum latency value. If performance is sufficiently important that it can threaten safety, it is reasonable to assume the architect will be able to estimate the required latency.

The target reasoning framework is applied and the scenario is evaluated. As is usually the case, the reasoning framework computes an estimate for the quality using the current architecture and determines whether that value satisfies (We use the term “satisfice” since the outcome of the analysis does not tell us the optimal solution to satisfy the safety requirements but instead it tells us that the scenarios have reached their minimum allowed thresholds.) the derived scenario by estimating whether the quality threshold would be reached. That judgment is returned to the safety reasoning framework as input to the determination of whether the safety scenario is satisfied.

When multiple quality attributes are identified for a safety scenario then the results from all of these attributes are combined to determine whether the safety scenario has been satisfied. For example, in Figure 2, a safety scenario is composed of two other scenarios which are related to usability and confidentiality. Those two other scenarios would both have to be satisfied in order to satisfy the corresponding safety scenario.

Once the individual safety scenarios have been evaluated, the architect checks the set of safety scenarios to see which have been satisfied. The priority is based on the cost of an accident and the probability of the hazard causing an accident. The architect changes the architecture in response to the values returned by the reasoning framework. The evaluation process is repeated until, in the expert judgement of the architect, a sufficient number of the scenarios are satisfied.

### 3.4 Analytic Constraints

The main limitations on the safety reasoning framework are the focus on qualities and the dependence on the architecture representation. There are failures that are due to functional problems that are not the result of a failure to satisfy a quality requirement [11]. The reasoning framework does not address these problems since they are largely outside the software. For each safety scenario, the quality attribute estimate computed by the reasoning framework depends on the network of responsibilities that represent the system. The completeness, correctness, and consistency of this network determines the quality of the network.

### 3.5 Model Representation

The model representation for the safety reasoning framework is a network of reasoning frameworks. The mapping that transforms the individual safety scenarios into scenario(s) in the appropriate reasoning framework(s) constructs the model. It is the set of arrows shown in Figure 2 that originate from the safety scenario. Constructing the mapping requires the architect to identify the qualities represented in the safety scenario and then to provide the additional information that is needed for the target reasoning framework that is not used by the safety reasoning framework.

The resulting model is a network of reasoning frameworks that are joined together by transformations and returned results. This is made possible by considering each reasoning framework to be a view on a single architecture. In tools such as ArchE the architecture is a network of responsibilities. Different reasoning frameworks add their own linkages between responsibilities, but the set of responsibilities remains the same.
3.6 Interpretation

The hazards obtained by FHA are subjected to FTA to identify their respective causes and the quality attributes behind them. The hazards are prioritized based on criticality as described in section 2. These hazards are associated with their respective quality attributes (causes). The safety scenarios associated with each hazard are mapped to the scenarios of the quality attributes related to the corresponding hazards. Figure 2 gives a conceptual view of the mapping. In this example, the safety scenario is mapped to usability and confidentiality scenarios.

3.7 Evaluation Procedure

The quality attribute scenarios that are derived from the safety scenarios are each evaluated in the appropriate reasoning framework. The result of that evaluation is locally judged to have satisfied its requirement. The data available to the safety reasoning framework are boolean, satisfied/not satisfied, judgements for each scenario.

We exploit the risk-based nature of hazards to rank the hazards identified in the initial safety analysis. This approach uses the probability that the hazard will occur, the probability that the hazard will lead to an accident if it does occur, and the cost if there is an accident to determine the ranking. The ranking is an ordered list of satisfied and not satisfied scenarios. There is no intention to derive a single score from the set of safety scenarios.

The ranking of the scenarios gives the architect an order in which to attack the scenarios that have not satisfied their requirements. Suggestions regarding how to improve the safety aspect of the architecture can be made to the architect using tools that draw on the reasoning frameworks for the derived scenarios.

One aspect of the qualitative nature of the safety reasoning framework shows up when we think of which transformations to apply. Certain tactics will improve some scenarios and diminish others (based on general trade-offs) [7] and by applying these tactics based on that general observation (a qualitative method) we may be able to satisfy all the safety scenarios (or not).

3.8 Summary

Our technique fits well within the typical iterative software architecture design process. The architect makes changes to the network as a result of the feedback from all of the reasoning frameworks. The individual reasoning frameworks re-evaluate the status of each scenario and report back an update on whether each scenario is satisfied by the new architecture.

4 Example Application of Technique

We will illustrate our technique using the Clemson Travel Assistance System (CTAS) [13]. It is an itinerary planning system that plans routes and modes of transportation from one place to another. CTAS can be executed on various platforms including handheld devices. The hand-held devices add GPS and use wireless connections to check schedules, make reservations, and check the status of various modes of transportation. Developers can develop plug-ins that integrate appropriate technologies to provide various services through CTAS. The quality attribute requirements of CTAS include performance, security, functionality, reliability, usability, safety, efficiency, maintainability and portability [13]. With these drivers in mind, the architecture representation of CTAS, shown in Figure 4 will be used to apply our technique of safety analysis and transformation. Due to space limitations we show only the most pertinent results.
4.1 Perform Initial Safety Analyses

The architect was assisted by domain experts to define the context in which CTAS operates and to identify potential failures. Table 4 shows a portion of the results of the FHA on CTAS.

The FHA revealed hazards associated with the itinerary management component, the user data management component, and the user interface component that could lead to safety problems. The next step in our analysis was to use each hazard as the starting node of a fault tree and analyze the root cause of the hazard. Figure 3 illustrates the fault tree for the hazards associated with the user data management component.

In the FTA diagram, the bold circle indicates a safety fault that is related to a quality attribute of CTAS. The other circles indicate safety faults that are related to the functionality of CTAS.

4.2 Translate into Safety Scenarios

The faults from the FTA that affect certain quality attributes were turned into safety scenarios. Using the general scenario for safety we constructed the concrete scenarios. We will only show the safety scenario mapped to the confidentiality scenario [6]. In this scenario, CTAS stores sensitive information such as address, telephone, social security number of the user, which could be used to harm the user. An unauthorized user accesses the user’s personal data. The data is used to locate the user’s home and the user is attacked at their home. The concrete scenario, in the standard form of the general scenario for the safety reasoning framework, is illustrated in Table 4.

<table>
<thead>
<tr>
<th>Stimulus</th>
<th>Access of confidential data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source of the Stimulus</td>
<td>Unauthorized user</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal mode</td>
</tr>
<tr>
<td>Artifact</td>
<td>Personal Data of user</td>
</tr>
<tr>
<td>Response</td>
<td>End user’s personal data is accessed</td>
</tr>
<tr>
<td>Response Measure</td>
<td>The loss of privacy the user experiences due to the unauthorized access</td>
</tr>
</tbody>
</table>

Table 4: Safety Scenario related to potential confidentiality failure

<table>
<thead>
<tr>
<th>Stimulus</th>
<th>Attempt to read CTAS user’s social security number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source of the Stimulus</td>
<td>Unauthorized person</td>
</tr>
<tr>
<td>Environment</td>
<td>End user’s hand-held CTAS device</td>
</tr>
<tr>
<td>Artifact</td>
<td>CTAS database</td>
</tr>
<tr>
<td>Response</td>
<td>The social security number is read</td>
</tr>
<tr>
<td>Response Measure</td>
<td>The amount of physical harm that comes to the user whose SS number was read</td>
</tr>
</tbody>
</table>

Table 5: Confidentiality scenario after mapping from the safety scenario

4.3 Apply the Constituent Reasoning Frameworks

Each of the safety scenarios was mapped to the specific reasoning frameworks corresponding to the derived qualities, as illustrated in Figure 2. In other words, each scenario was transformed into the form prescribed by the general scenario for that framework. We show the process only for the confidentiality scenario. The general safety scenario was mapped to the general confidentiality scenario. The confidentiality scenario needed the architect to identify the data that was to be safeguarded, which actors in the use case model were the unauthorized users, and the amount of exposure that was permissible. The safety scenario in Table 4 was mapped to the confidentiality scenario in Table 5. The mapping was done to a confidentiality scenario because of the keyword “unauthorized” found in the safety scenario. The architect provided the Stimulus, Response and Response Measure goal for the new scenario.

Table 5 shows the concrete confidentiality scenario that resulted from mapping the safety scenario to the confidentiality reasoning framework. The actual Response Measure when evaluated by the confidentiality reasoning framework is “failed” for the current CTAS architecture since some sensitive data was accessed by an unauthorized person. The result that the scenario is not satisfied is returned to the safety
reasoning framework as the result of the confidentiality reasoning framework.

4.4 Apply the Safety Reasoning Framework

The architect determined to what extent the safety scenarios have been satisfied by the current architecture. When all of the safety scenarios have been evaluated in the constituent reasoning frameworks the architect had a set of true/false results. In our analysis, the confidentiality and the usability scenarios were not satisficed by the current architecture so the corresponding safety scenarios were not satisficed. The architecture needed to be changed to satisfice the two scenarios under consideration.

The architect used the criticality among the scenarios, established early in the initial safety analyses, to determine the order in which transformations are going to be applied to the architecture. In our example, the architect has previously determined that the confidentiality hazard takes precedence over the usability hazard. The architect followed suggestions generated by the confidentiality and usability frameworks and made modifications to the architecture.

The original architecture, as a set of responsibilities in the form of an architectural cartoon, is shown in Figure 4. The dashed lines indicate responsibilities that have been added as transformations to the architecture.

In order to improve the confidentiality scenario, an authentication responsibility was added. These transformations in the architecture are not conflicting; however, the realization of the authentication responsibility did reduce the usability of the system. A voice output responsibility was added to improve the usability scenario. The revised architecture now satisfies the confidentiality and usability scenarios which implies that the two safety requirements we are investigating are both satisficed.

By satisficing the two safety scenarios, a determination based on the safety goals of CTAS can be made. And from the analysis we have determined that the safety scenarios related to usability and confidentiality have been satisficed.

The result of the analysis are expressed using a star plot was shown in figure 5. The various levels on the star plot indicate the degree to which a quality attribute is satisficed (0 - Unsatisficed, 1 - Minimum level satisficed, 2 - Good level satisficed, 3 - Max level satisficed). The value on each axis is a weighted average calculated from the the scenarios found in the analysis of that quality attribute. We omit the confidence intervals for each analysis because of space limitations.

The star plot is also used to determine to what extent an architecture has become safer after certain transformations have been applied. The typical approach is for the architect to use the star plot to guide modifications to the architecture.

5 Related Work

Much work has been done to develop methods for satisfying software safety requirements. At the architectural level, techniques such as Failure Propagation and Transformation Notation (FPTN) [4] and Component Fault Trees (CFTs) [8] are used to evaluate whether a software architecture can meet the safety requirements. These approaches are based on modular failure propagation models that model the system in terms of components that create, transform and consume failures [5]. These techniques allow the architect to determine the likelihood that a hazard would oc-
cur based on the software architecture. Our technique complements existing techniques by taking a broader perspective of safety that includes quality attributes.

6 Conclusion and Future Work

We have demonstrated the viability of implementing a quality attribute reasoning framework by decomposing it into other reasoning frameworks. The mappings used in this work transform a safety scenario with a required level of safety into scenario(s) in the target framework(s) with corresponding levels of those attributes necessary to satisfy the original safety requirement. The results in the target reasoning frameworks provide the data for the safety reasoning framework to use to guide the architect toward beneficial modifications of the architecture.

Our work on reasoning frameworks has led us to identify relationships among frameworks. Initially we found that one way to address the varying definitions of an attribute is to view that attribute as a composite of other attributes. For example, an attribute such as dependability is really an aggregation of other attributes including reliability, availability, confidentiality, integrity, maintainability and safety. Now our work on safety has led to the realization that safety, as a quality, is a facade that has dependencies on other attributes for analytic theories that provide estimates of attribute values. Our future work will explore these relationships more deeply and possibly identify other relationships as well.

References


