Managing Architectural Design Decisions for Safety-Critical Software Systems

Weihang Wu and Tim Kelly

Department of Computer Science, University of York, York YO10 5DD
{Weihang.Wu, Tim.Kelly}@cs.york.ac.uk

Abstract. In this paper, we propose a negative scenario framework along with a mitigation action model as the linkage between safety quality attribute and architecture definition. The scenario framework provides an effective means of formulating safety concerns. The mitigation action model facilitates exploitation and codification of existing safety-critical system design knowledge. Finally, we present a series of steps that enable the justification of architectural design decisions that refine both requirements and architectures. We demonstrate and discuss the application of our framework by means of a case study.

1 Introduction

Over the last decade, the importance of quality requirements (characterised by quality attributes) has been well recognised in architectural design. Recently, the SEI[1] has established a design methodology termed *Attributed-Driven Design* (ADD) [6] to emphasise the active role of quality attributes (QAs) in architecture design. Quality attributes can be achieved by a set of design options ranging from coarse-grained design patterns to more finer-grained design techniques. The success of the design of an architecture thus lies with judicious decisions made when selecting the appropriate design options and composing the selected options into the architecture. Among quality attributes, a great significance has been given to software risks in the safety-critical system domain. Examples of the most serious computer-related accidents in the past 20 years such as Therac-25 [12] and Ariane 5 [13] can be attributed to flawed system and software architectures. However, existing practice fails to systematise architectural design approaches to addressing these safety concerns.

In this paper, we examine the factors involved in making the principled choices within a safety-related architectural design space and present an approach to rationalising the architectural decisions behind the selection and the impacts on both requirements specification and architectural modelling. The paper is organised in the following sections. Section 2 introduces the key concept of the "safety" quality attribute and existing practice in architectural design for safety. Section 3 presents our framework for decision making. Section 4 demonstrates this framework by means of a case study. Section 5 discusses preliminary findings and related work. Finally, section 6 presents concluding remarks.

[1] Software Engineering Institute at Carnegie Mellon University.

C. Hofmeister et al. (Eds.): QoSA 2006, LNCS 4214, pp. 59 – 77, 2006.
© Springer-Verlag Berlin Heidelberg 2006

2 Safety and Architecture

Safety is freedom from accidents or losses; software safety implies the contribution of software to safety in its system context [12]. The remainder of this section analyses the concept of safety as a quality attribute in terms of its underlying concerns and discusses existing solutions and their limitations.

2.1 Safety as a Quality Attribute

Like security, at the heart of defining the safety attribute is the identification and evaluation of *negative* requirements such as unwanted or unplanned events and conditions. Typically, the safety lifecycle [2] starts by identifying negative requirements over existing system-level context. Safety requirements are then derived by choosing appropriate mitigation mechanisms to protect against the negative requirements identified. These safety requirements will in turn act as the constraints on the system and software design. Chosen mitigations may themselves bring new safety problems. New mitigations may thus need to be identified and safety requirements refined. The problem is complicated by the fact that the details of the system are often unavailable until the late stages of development. Hence, a key question is how to elicit and formulate the negative requirements along with the corresponding mitigation mechanisms in co-ordination with the architectural design process in an evolutionary manner.

Another vital aspect of safety is risk. From an engineering standpoint, there is no such thing as absolute safety. Safety is often defined as the measure of the degree of the freedom of risk under all conditions [12]. The basic tenet of evaluating safety-critical systems is thus to ensure that these systems present an acceptable level of risk balanced with their cost. Therefore, an important question is how to integrate risk assessment and cost benefit analysis with the safety design process.

Finally, safety is an emergent system property and it is not possible to take a single system component such as a software module in isolation to assess its safety. In practice, the emergent safety properties are enforced by a set of safety constraints upon the architecture. At the top level of design, these safety constraints are imposed upon the whole system (i.e., the system under design and its environment) and often expressed in an absolute functional form (e.g., ‘must’ or ‘must not’). As the design process progresses, the safety constraints are eventually refined into other quality requirements such as performance and availability targets allocated onto the behaviours of relevant architectural components. Inevitably, our concern is thus how to capture such a linkage between safety and non-safety qualities.

2.2 Design for Safety

A large number of design techniques for safety have emerged in both research and practice. Safety is thus achieved by deciding upon the appropriate design techniques to be employed in a specific system context. In general, current practice advocates two classes of design approaches: