Chapter 10
Supporting Quality-Driven Software Design through Intelligent Assistants

Alvaro Soria
ISISTAN Research Institute and CONICET, Argentina

J. Andres Diaz-Pace
Software Engineering Institute, USA

Len Bass
Software Engineering Institute, USA

Felix Bachmann
Software Engineering Institute, USA

Marcelo Campo
ISISTAN Research Institute and CONICET, Argentina

ABSTRACT

Software design decisions are usually made at early stages but have far-reaching effects regarding system organization, quality, and cost. When doing design, developers apply their technical knowledge to decide among multiple solutions, seeking a reasonable balance between functional and quality-attribute requirements. Due to the complexity of this exploration, the resulting solutions are often more a matter of developer's experience than of systematic reasoning. It is argued that AI-based tools can assist developers to search the design space more effectively. In this chapter, the authors take a software design approach driven by quality attributes, and then present two tools that have been specifically developed to support that approach. The first tool is an assistant for exploring architectural models, while the second tool is an assistant for the refinement of architectural models into object-oriented models. Furthermore, the authors show an example of how these design assistants are combined in a tool chain, in order to ensure that the main quality attributes are preserved across the design process.

DOI: 10.4018/978-1-60566-758-4.ch010

INTRODUCTION

Software design can be seen as the bridge between requirements and implementation. Over the last years, design has become a central practice in software development, mainly due to the growing complexity of today's systems and the impact of *quality attributes* in software products. Quality attributes capture non-functional concerns such as: performance, reliability, security, or modifiability, among others (Bass et al., 2003). Along this line, architecture-centric design approaches are being more and more adopted, since they help to explicitly engineer quality rather than considering it an afterthought. In these approaches, the role of the designer is to plan for a design solution that is "good enough" for the competing interests of the stakeholders. The designer must make decisions for architectural patterns and gross decomposition of functionality as early as possible in the development cycle, in order to reason about the advantages and disadvantages of potential design solutions. There are usually multiple solutions that satisfy the same requirements, and each of these solutions is likely to have tradeoffs regarding quality attributes (Boehm & In, 1996). The notion of *tradeoff* means that the improvement of one quality comes at the cost of degrading another, as it is the case of modifiability versus performance. Therefore, creating a design that meets a set of quality-attribute requirements is a difficult and challenging problem, even for experienced developers. We see here an interesting area for the application of automated design assistance.

According to their granularity, two types of design activities can be identified: *architecture design* and *detailed design*. Architecture design deals with the high-level organization of the system in terms of components, relationships between these components and allocation of functionality. This organization is generally known as the software architecture of the system (Bass et al., 2003). The *software architecture* is the primary carrier of quality attributes for a system because it prescribes how the system should be realized by concrete implementations. Once an architecture exists, detailed design is about the decisions related to the computational implementation of that architecture (e.g., code, existing libraries and components, frameworks, etc.). A common choice for detailed design and implementation is the object-oriented paradigm (Rumbaugh et al., 1991).

Both architecture design and detailed design require designers to apply their technical knowledge and experience to evaluate alternative solutions before making commitments to a definite solution. Normally, a designer starts with a guess of the solution, and then goes back and forth exploring candidate design transformations until arriving to the desired solution (Tekinerdogan, 2000). We conceptualize this exploration of the design space into two main phases: (i) from quality-attribute requirements to (one or more) architectural models - called *QAR-to-AM phase*, and (ii) from an architectural model to (one or more) object-oriented models - called *AM-to-OOM phase*. Making the right design decisions for each phase is a complex, time-consuming and error-prone activity for designers. Although tools for specification and analysis of designs exist, these tools do not support the designer in making informed decisions based on quality-attribute considerations. Along this line, several AI developments have shown the benefits of improving conventional tools with intelligent agents. The metaphor here is that the agent acts like a *personal assistant to the user* (Maes, 1994). This assistant should be able to monitor the designer's work, and offer timely guidance on how to carry out design tasks or even perform routine computations on her behalf. For example, given a modifiability scenario, a design assistant could recommend the use of a Client-Server pattern to satisfy that scenario. If the designer agrees to apply such a pattern, the assistant could also take over the assignment of responsibilities to Client and Server components. Also, the assistant could remind the designer to verify the influences of the