Improving quality attributes of software systems through software architecture patterns

Harrison, Neil Bruce

Summary

Some of the most important requirements of any system are those that concern its quality attributes, such as reliability, performance, security, and usability. Quality attributes tend to be system-wide characteristics. Because of this, they require system-wide design approaches; that is, measures to achieve these quality attributes have architectural implications. Conversely, architectural decisions can make it easier or harder to achieve the quality attributes. Thus early decisions impact the quality attributes.

During the process of software architectural design, architects select architecture patterns to be used. The structure of the patterns is clear, but their impact on the all-important quality attributes is not apparent from observing the patterns. That means that one might select patterns that are incompatible with or even detrimental to the satisfaction of quality attribute requirements. However, one cannot easily ascertain whether a quality attribute is sufficiently satisfied until the system is largely complete. By this time, architectural changes can cause significant disruption to the code already written. In short, architects must select architecture patterns early, yet the impact of these decisions on the quality attributes are not fully understood until it is too late to easily change the architecture.

The consequence of this problem is that software may fail to meet its quality attribute requirements. The consequences can be serious: at the very least, users may experience inconvenience and frustration. Beyond inconvenience, software quality attribute failures may burden users and software providers with financial liabilities.

The goal of this research is to leverage patterns to create architectures that meet quality attribute requirements, during the analysis, synthesis, and evaluation phases of software architecture. This will help improve the ability of architects to design systems that meet their quality attributes requirements; helping them to make informed decisions at architectural design time about how well the software under design will satisfy the quality attributes. This can be done because architecture patterns impact quality attributes in regular ways, and we can understand the nature of these interactions. The knowledge of these interactions can be applied to the architectural design and evaluation processes.

This work begins by exploring how extensively architecture patterns are used, as well as which patterns are most commonly used. It was determined that architecture patterns are very common, found in nearly all industrial systems. Most software systems use between 1 and 4 architecture patterns. The most commonly used architecture patterns are, in descending order of frequency, Layers, Shared Repository, Pipes and Filters, Client-Server, Broker, Model View Controller, and Presentation Abstraction Control.

This leads us to the question of how architecture patterns' use impacts important quality attributes of systems. The application of a pattern constitutes a major

architecture decision that achieves the satisfaction of a quality attribute requirement or the lack thereof.
Architectural decisions are important artifacts of architectural knowledge, as they help future developers understand the architectural structure and the reasons particular structures were originally used. Architecture patterns embody major architectural decisions about which architectural structures to employ. Because architecture patterns are well understood and documented, use of architecture patterns helps solve the difficult problem of documenting architectural decisions, the rationale behind them, and their consequences, particularly with respect to quality attributes.

Different patterns have different impacts on quality attributes; thus certain patterns and quality attributes are compatible with each other, while others are less so. Among the most common quality attributes, usability is highly compatible with Model View Controller, security with Broker, maintainability and reliability with Layers, efficiency and portability with Pipes and Filters, and implementability with Broker.

The nature of quality attributes influences their compatibility with architecture patterns. Quality attributes are achieved through the implementation of specific measures called runtime tactics. The tactics are implemented within the structure and behavior imposed by the architecture patterns, and may require some measure of change to the pattern. The types of significant changes to patterns include adding components that do not fit with the pattern, or changing the nature of connections among components of the pattern. The magnitude of change caused to a pattern by the implementation of a tactic can thus be evaluated.

Because nearly all systems employ multiple patterns and are concerned with multiple quality attributes, the impact on the architecture of implementing a tactic is somewhat more involved. Further analysis of tactics reveals that some naturally impact all components in a system, while others can be implemented in just a few selected components. If the tactic can be implemented in just a few components, it might be implemented within the architecture pattern where it best fits. However, the decision of where it is implemented is strongly influenced by the system requirements; this takes precedence over goodness of fit with the patterns.

The information about how tactics interact with patterns can be used during the processes of architectural evaluation and synthesis to help architects create architectures that more easily support the quality attributes. They can help architects select among alternative tactics for a particular quality attribute, and choose among different patterns to use. This can be done as part of normal architecting processes.

This information is particularly useful during architectural evaluation activities. It can help reviewers of architectures identify potential problems in architectures where the current architecture may hinder the effective implementation of tactics for the desired quality attributes. It can also identify potential opportunities where

additional or alternate patterns or tactics may be employed. Numerous pattern-based architecture reviews have been completed, and have demonstrated their utility. In nearly every case, significant architectural issues were uncovered, resulting in improvement to the architecture.