Chapter 5

Architectural development methodology

5.1 Introduction

In this chapter, we compare two different software architecture development methodologies, namely the developed by the Software Engineering Institute (SEI) of Carnegie Mellon University Attribute Driven-Design (ADD) [7] and the suggested by Microsoft technique for architecture and design [8], and justify the decision for usage of the ADD method for system design of the developed project. Additionally, we outline that this decision has been made based on the specifics of this project and its system's requirements.

5.2 Overview of the Attribute Driven-Design

Attribute Driven-Design (ADD) is a process used for development of architectural design of software systems. It is an iterative process with each iteration involving multiple stages. In this approach, the process of development of software architecture design is based on the requirements of the architecture. At the initial stage of the ADD process the requirements are identified, categorised, and rated. There are three main categories of requirements namely Design Constraints, Functional Requirements, and Quality Attribute Requirements. The rating is related to the impact of the requirement on the architecture and the importance of the requirement to the architectural stakeholders. After the requirements with the highest ratings are identified the system is recursively decomposed, in accordance with the requirements selected. At each stage of the decomposition architectural tactics and patterns are considered and those which most closely meet the requirements of the quality attribute scenarios are selected [7].

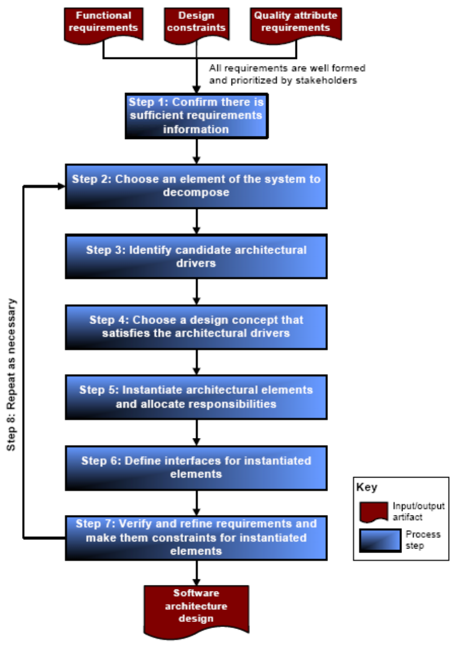
Functional requirements
Design constraints
Quality attribute requirements
All requirements are well formed and prioritized by stakeholders
Step 1: Confirm there is sufficient requirements information
Step 2: Choose an element of the system to decompose
Step 3: Identify candidate architectural drivers
Step 4: Choose a design concept that satisfies the architectural drivers
Step 5: Instantiate architectural elements and allocate responsibilities
Step 6: Define interfaces for instantiated elements
Step 7: Verify and refine requirements and make them constraints for instantiated elements
Step 8: Repeat as necessary
Software architecture design
Key
Input/output artifact
Process step

Figure 5.1: Steps of ADD [45].

Overall the ADD recursive method comprises of two parts: part 1 - usage of tactics to achieve quality requirements and part 2 – documentation of the decomposition [7]. ADD is a top-down method where the designers are examining the whole system at first and drill down into its elements. This approach is useful because it allows the designers to identify the quality attribute trade-offs at the early design stages and leads to architectural design which satisfies both the quality and functional requirements of the system.

5.3 Overview of the Microsoft technique for architecture and design

Microsoft suggest an iterative and incremental technique for development of software architecture design. It comprises of 5 steps:

1. Identification of Architecture Objectives
2. Identification of key scenarios
3. Creation of application overview
4. Identification of key issues

5. Definition of candidate solutions

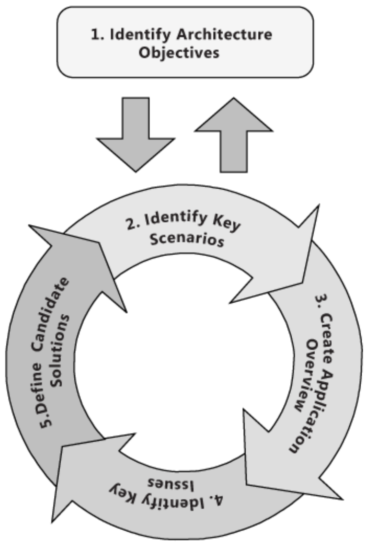
1. Identify Architecture Objectives
2. Identify Key Scenarios
3. Create Application Overview
4. Identify Key Issues
5.Define Candidate Solutions

Figure 5.2: The iterative steps for core architecture design activities [8].

This technique helps the software architects produce an architecture based on the requirements regarded as most important such as business critical requirements and requirements having high impact such as security requirements. After the first iteration, a base line architecture is produced. It is a high-level design which, after testing against key scenarios, quality attributes, known constrains, and requirements, in the consecutive iterations is refined and more details are added to it. The baseline architecture is used for building of candidate architectures. Architectural spikes, or with other words software implementation of small part of the overall design or architecture, could be used for testing the candidate architectures, exploration of specific areas, and validation of new concepts.

This approach, especially in the agile software development context, suggests that each iteration comprises of both architectural and development activities. That way the big up front architectural design approach is avoided. Additionally, in this approach, each iteration is carried on vertical slices of the application rather on horizontal meaning that only architectural elements that are required for the selected architecture objectives are designed and developed rather the functionality for an entire layer of the application. Thus, the architecture could be produced faster and could be validated by the users before a great deal of functionalities are developed.

5.4 Conclusion

It is worth mentioning that there are many more methodologies for development of software architecture design neither of which being better of worst. The degree of applicability of the different methodologies depends on a great number of factors including software development methodology, the application domain and purpose, and preferences and experience of the software architecture stakeholders. For example, a business application would benefit more from agile software development process and architectural design process, avoiding the big up-front design, such as the one suggested by Microsoft since the business environment is changing at rapid rates and the users find new strategies and form new requirements as they use the application. On the other hand, a mission critical system such as an air traffic control system would benefit more from planned software development and detailed up frond software architecture design such as the one provided by ADD. ADD might take many iterations and time before the actual software development commences but it allows the designers to identify the quality attribute trade-offs at the early design stages and leads to architectural design which satisfies both the quality and functional requirements of the system.

For the purposes of this master's dissertation project an extensive research was required. As a result, all of the applications requirements where defined before the start of the software architecture design and it was identified that changes to the requirements during development of the software architecture design were not expected. For this reason, we selected the Attribute Driven-Design method for development of software architecture design.