Implementation and Evaluation of Mobile Pesuasive Personalized Food Recommender System

Muhammad Kabir Khan

April 12, 2015

Contents

Chapter 1

Introduction

This chapter will provide you with the brief introduction of the topic in question. The motivation of this research project, goals and the related work will be discussed in detail.

1.1 Motivation

In engineering sciences, the term *Architecture* has great significance. Similarly, if we talk about Software Engineering in particular, the importance of *Software Architecture* cannot be neglected. It becomes an important activity in software development life cycle. In this modern era of complex software systems, the design and overall structure (Software Architecture) of a system are more significant aspects than the choice of algorithms and the data structures.

Designing the architecture of a software system is not sufficient, but there is one more important thing, which needs to be taken into consideration that Software Architecture is fully implemented? or Code actually adheres to the targeted architectural model? In practice, we have geographically distributed teams, located at different parts of the world, so there is always a *Model-Code gap*, which affects the conformance of the architecture. There exist inconsistencies between the model and the code, which are made accidentally or through the evolution of either of these.

1.2 Problem Statement

In the above presented scenario, Software architecture which is mainly specified using the Unified Modeling Language (UML) by architecture teams is handed over to implementation teams. During implementation, architecture is then violated, accidentally or not, by the implementation teams. In this way we have architecture violations in form of inconsistencies. These inconsistencies are also introduced, if either of these model or code evolve. For example, code evolves as result of added features or fixing of bugs. Model evolves in response to business planning needs.

If we run an implementation compliance analysis tool which detects the architecture violation, we get a lot of findings. Now, based on these findings, we need to introduce a *Tolerated Model*, a mechanism for the prioritization of these violations (acceptable, not acceptable and critical) with respect to their level of severity. Some of these violations are tolerable and can be ignored, but some violations though trivial, can result in system failure. With help of this tolerated model, we will review these violations and provide feedback to the developer or architect in order to remove the violation.

1.3 Goals

The goal of this thesis is to find a way to prioritize architecture violations using findings from implementation compliance analysis tool.
In order to achieve this goal, we need to know:

- **What is a dependency in software architecture?**
  Do some literature work in order to understand the term Dependency in context of Software Architecture.
- **Which dependencies are important for the practitioners?**
  Conduct interviews with practitioners at Intel to ask:
  - What is a Dependency?
  - Which Dependencies are relevant?
  - How to prioritize them?
- **How to prioritize these dependencies in a real world environment?**
  Create a Tolerated dependency model Create a review module for the dependency violations

1.4 Contributions

Chapter 2

Theoretical Background

2.1 Important Foundations

2.1.1 Software Architecture

During the last few years, complexity in modern software has increased dramatically. With this increased complexity and distributed nature of software, the power and importance of Software Architecture can be realized. Software Architecture is an Art, an art of structuring the software in terms of modules, components, services and layers, that makes it scalable, adaptable, maintainable and change tolerant.

Architecture is defined by the recommended practice as the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution [1]

Software Architecture depicts the high level or abstract picture of the software. It makes lot easier for the architects to make design decisions, and identify critical issues at an earlier stage. A good architecture has a well organized structure, which reduces the complexity and leads to better understanding of the system. Once we have achieved it, we can fulfill all the requirements.

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. [2]

2.1.2 Software Architecture Conformance Analysis

The quality attributes such as performance, security, modifiability and reliability depend heavily on the architecture of the software. Quality is achieved if the implementation conforms to the constraints and design principles of the architecture. But unfortunately, the implementation never realizes the abstractions of the architecture completely. Often, there is a sizable gap between these two. This divergence can be due:

- Negligence of the developer
- Lack of documentation for the design artifacts
- Architecture flaw identified during implementation
- Natural evolution of the model/code

Architecture conformance analysis includes processes and strategies, whose intent is to remove or reduce the level of divergence and keep the architecture sync with the implementation or vice versa.

2.1.3 Architecture Violation

2.1.4 Software Dependency

Software Dependency is a system level concept for measuring the independence of the system components. Increase in dependencies results in increased coupling between the system components which exhibits more defects than lower dependencies.

Dependency is a relational concept between two entities (System, Components, Modules, Classes or Functions). This relationship can either be Functional Dependency i.e. function A calls function B or it can be Data-Related Dependency i.e. a data structure is being utilized in one function and modified in another.

- Compile and Run-time dependencies
- Visible and Hidden Dependencies
- Direct and Indirect Dependencies
- Local and Context Dependencies

2.1.5 Model-Code Gap

2.2 Related Work

Chapter 3

Prioritization of Architecture Conformance Findings

3.1 Pre-Study

3.2 Approach

Chapter 4

Discussion

Chapter 5

Conclusion

5.1 Summary

5.2 Future Work

Bibliography

[1] ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems

[2] L.Bass, P.Clements, R.Kazman, Software Architecture in Practice (2nd edition) Addison-Wesley 2003

List of Figures