Pattern-Based Model Refactoring for Context Awareness in Mobile Software

Seonghye Yoon, Deokyoon Ko, Soojin Park

Dept. of Comp. Sci. & Eng., Sogang University, Seoul, South Korea
seonghye@sogang.ac.kr, maniara@sogang.ac.kr
Graduate School of MO, Sogang University, Seoul, South Korea
psjdream@sogang.ac.kr

Abstract. User requirements for mobile software are maturing along with the industry, with request for quality attributes growing in diversity. The quality of context-awareness, recommending appropriate services based on user contexts, have become of particular importance. Satisfying new user demands requires mobile software developers to transit to writing adaptive software. Unfortunately, up to now this transition has not been very successful, and developers have been limited to making trial attempts with adaptation frameworks, mainly due to insufficient understanding and experience with adaptive software. This paper presents a guideline for identifying context awareness factors in mobile software and proposes a pattern-based model refactoring method to facilitate efficient transformation from traditional to adaptive software.

Keywords: Re-engineering, Mobile Systems, Adaptive Software

1 Introduction

The emergence of smartphone have resulted in rapid increase in mobile software size and complexity, while the success of application markets have stimulated various services to be released in the form of mobile applications[1]. Demands from smartphone users are accordingly growing, not only for new features but also towards various qualities attributes, and among the highly sought-after is the quality of context-awareness. Existing mobile software typically perceives its context within a fixed logic and executes fixed behavior, thereby limited in its flexibility to respond to various environmental changes.

The characteristic expectation for software to perceive and adapt to various environments in which it is executed is similar to the characteristics of adaptive software[2]. Adaptive software separates its adaptation conditions and behaviors from its main module using various adaptation mechanisms[3][4] to facilitate behavioral expansion or modification. Transforming existing mobile software into adaptive mobile software via implementing additional behaviors as separate modules and defining the conditions for module execution can enable the software to continuously adapt to changing environments.
While research on adaptive software is vigorously ongoing, mobile software developers still find their understanding and experience of adaptive software insufficient. To resolve this issue, more in-depth research on transforming existing software to adaptive software is called for. In [5], a reengineering method to support reliable dynamic adaptation using model-based methods is proposed. While the research provides the essential mechanisms and processes, it is limited in scope in that guidelines for identifying the conditions and resulting behaviors of context awareness are not defined and left for developers to independently determine, thereby limiting real-life application.

In this paper, to solve the problem of difficult-to-identify adaptation elements, we propose a pattern-based model reengineering method which incorporates context awareness factors as to modify existing software to adaptive software. First, to guide adaptive behavior criteria, context awareness factors provided by the android framework are automatically identified on the API level. Then the conditions directly affecting context awareness factors and their resulting behaviors are referenced as to apply appropriate adaptation strategy pattern to the source code, thereafter constructing a source code skeleton as to support adaptive mobile software development.

The remainder of this paper is as follows: Chapter 2 describes the proposed pattern-based model refactoring method for supporting context awareness in mobile software, and Chapter 3 summarizes the paper with directions for our future research.

2 Pattern-Based Model Refactoring Method

Our proposed method employs a four-step process to transform non-adaptive software to adaptive software, as depicted in Figure 1. The steps are: 1) automatically identify adaptation trigger code from the source code, 2) select appropriate adaptation strategy pattern applicable depending on the conditions of identified adaptation trigger code and the desired additional behaviors, 3) apply the selected patterns to the source code and refactor it into an adapted model, and 4) generate adapted source code using the adapted model.
2.1. Detecting adaptation trigger code

The first step of our proposed process is to automatically detect adaptation trigger code which should provide context aware services. The factors which trigger the adaptation trigger code is searched for using domain knowledge composed of 1) understanding of events passed on from Android framework to the software, and 2) APIs which calls objects potentially influenced by the aforementioned events. For example, events such as GPS_EVENT_STARTED generated at starting the GPS function, or onLocationChanged() method call on the Location listener waiting for GPS status change, can be identified as factors of adaptation trigger code. If an identified factor is determined to be a condition to trigger behavioral changes, then a variation flag is set for the particular code.

2.2. Selecting adaptization strategy patterns

In the pattern selection step, conditions which trigger adaptational variation and their resulting behaviors are analyzed per each variation flag set as to recommend appropriate adaptation strategy pattern for implementation. Each adaptation strategy pattern is composed of following elements:

- Problem: This element describes the environment or situation in which context awareness is required. The problem element details the relationship between with factors of adaptation trigger code and the pattern as to support appropriate recommendation.

- Solution: This element describes the methods of MAPE (monitoring, analyzing, planning, and executing) [6] context awareness, with adequate considerations for the following characteristics:
  - Fail-safe binding: Fail-safety becomes highest priority in case flexible plug-in support for external components or sensors is needed. Reliability must be ensured as to prevent the software from being influenced by unexpected absence of or misbehavior of external components or sensors. For example, if the GPS sensor has been disabled due to low battery, then GPS-dependent conditions should not be marked for adaptation as it may lead to unintended behavior or even crash of affected software.
  - Dynamic loading: When new adaptational variation points are added or new variants are added to existing variation points, those additions should be reflected back to run-time. For example, assume that software is developed in an environment which is a composition of Android framework and an OSGi (Open Service Gateway initiative) [7] framework providing dynamic update and component composition feature. In this case, the software’s non-main application modules participating in behavioral variations should be developed as OSGi bundles as to enable dynamic loading.
Rapid deployment: Since adaptive environment requires that new modules be created, updated and deleted, per-module installation and version management must be provided for adequate support.

- Participants: Participants can be categorized to be in the common area for the adaption framework or the variable area where variations are implemented. For example, assume that multiple modules provide identical features based on MUSIC[8] framework but varies in quality details. Then individual modules should be arranged in the manner that foundation classes negotiable in run-time environment through SLA Manager are placed in the common area, while classes providing individual functions are placed across a number of appropriate variable areas.

2.3. Applying adaptization strategy patterns to source model

An adapted model is generated through weaving selected adaptation strategy patterns to the source model. Considerations for conflicting participants must be given in this step, where conflicting participants are defined as participants with redundancy or contradictions due to multiple applications of (possibly identical) patterns. For example, if a particular pattern’s common area includes participants related to SLA manager, and the source model already has participants related to SLA manager, then those participants are viewed as conflicting. In such case, only the modified parts of the participants (methods, attributes, etc.) are considered applicable when weaving into the source model.

2.4. Generating adaptized source code skeleton

Adapted skeleton source code is obtained from the source code re-generated using the adapted model. It is left to the developer to assign variant conditions to the skeleton code and implement the functionality of the variants.

3 Conclusion

In this paper, we have proposed a pattern-based model refactoring method to facilitate the transformation from existing mobile software to adaptive software. The purpose of this research is to provide convenience and support for developers who have limited understanding or experience of adaptive software by automatically detecting adaption trigger code and then recommending appropriate adaptation strategy patterns for implementation. Adapted software modularized through patterns can achieve relatively higher maintainability and reliability than existing mobile software, and can support more efficient resource management through dynamic component loading in
run-time. Continued research in the future is to be directed towards building adaptation strategy pattern repositories to account for future mobile environments, and follow-on study will be conducted on verified pattern weaving methods with the goal of resolving potential conflicting participants.

Acknowledgements. This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (2012M3C4A7033348)

References