Chapter 11
Efficient Utilization of Patterns in Mobile Application Development:
An Architectural Model Approach

Jouni Markkula
University of Oulu, Finland

Oleksiy Mazhelis
Information Technologies Research Institute, University of Jyväskylä, Finland

ABSTRACT

A software pattern describes the core of the solution to a problem which tends to (re-)occur in a particular environment. Such patterns are commonly used as a means to facilitate the creation of architectural design satisfying the desired quality goals. In this chapter, the authors present first the challenges of applying patterns in mobile application development and give an overview of the patterns that are potentially useful in this domain. After that, they introduce a novel approach for organising patterns for mobile applications based on architectural model. In this approach, the identification of relevant patterns is considered as the process of reducing the set of candidate patterns by using constraints, such as required functionality and limitations of mobile terminals. Some of these constraints can be incorporated in a company-specific architectural model reflecting the commonalities in the solutions developed in the company. In this case, pattern lookup can be facilitated by cataloguing patterns according to the elements of this model. This approach has been validated with existing real company case application.

INTRODUCTION

The mobile technology domain is a challenging software design environment due to a number of technology-imposed and dynamically changing constraints, and the complexity of applied business models, among other factors. The complexity and peculiarities of the environment, as well as the rapid development of the technologies, introduce design problems which are not present in traditional application environments. This, in turn, often necessitates design solutions differing

DOI: 10.4018/978-1-61520-655-1.ch011

from those commonly applied in other conditions. These solutions should take into account the specifics of mobile terminals, such as limited battery power and small screen size, and the limitations of wireless networks, such as restricted bandwidth and availability. Furthermore, the applications should be future-proof, i.e. remain up-to-date in the face of quickly evolving technology, and they need to be implemented and ported quickly to new technologies and platforms. The solutions being developed need to meet a number of quality goals, such as modifiability, understandability, costs, and time to market, in addition to functionality and performance. Meeting these goals, to a large degree, depends on the architecture (Bass, Clements, & Kazman, 1999). Therefore, appropriate architectural design is of great importance in mobile application development.

Patterns have been introduced and adopted by the software community to support systematic design of architectural solutions with the desired qualities. A pattern in software engineering has been defined as a description of communicating objects and classes that are customized to solve a general design problem in a particular context (Gamma, Helm, Johnson, & Vlissides, 1995). Patterns should present a solution to a problem in a context, and their essential elements include (Gamma et al., 1995): The pattern name, the description of the design problem and its context, the solution describing elements, their responsibilities and collaborations, and the consequences of applying the patterns, e.g. the benefits and trade-offs. Patterns range in their abstraction level, varying from abstract architectural styles to design patterns and low-level programming idioms (Buschmann, Meunier, Rohnert, Sommerland, & Stal, 1996). They can be divided into general (or domain-independent) patterns (Gamma et al., 1995; Buschmann et al., 1996) and domain-oriented patterns tailored to specific areas of applications and/or to specific development tools or platforms, such as Core J2EE Patterns (Alur, Crupi, & Malks, 2001; Fowler, 2003).

Many of the patterns are potentially useful for application development in the mobile domain. The patterns applicable to the mobile domain can be divided into i) the patterns specific to (or particularly useful in) the mobile domain, and ii) the patterns from other domains or domain-independent patterns that can be also used in the mobile domain. Examples of mobile domain specific patterns are the Synchronization and the Remote Proxy patterns (Roth, 2002), as well as the Symbian two-phase construction and the cleanup stack idioms (Tasker, 2000). The patterns from other domains directly applicable in the mobile domain can be exemplified by Model-View-Controller (MVC) (Tasker, 2000) or by the Remote façade pattern (Yuan, 2004a; Fowler, 2003).

A large number of patterns have been elicited and distilled during last decade. Nowadays, patterns are available through a variety of media, e.g. books, journals articles and conference papers, internet catalogues and software framework documentation. However, finding the pattern particularly suitable in a specific problem setting has become more difficult. As a result, while the patterns document proven or successful solutions, the practical application of these patterns may be hindered, as finding a suitable pattern takes too long. The designer would merely avoid taking the risk of searching, learning, and applying a pattern (especially unknown beforehand), if considerable time needed to be spent locating the pattern.

Thus, the use of patterns promoting the development of flexible and reusable architectural and design solutions is vital for mobile software companies, and identifying essential and appropriate patterns for the mobile technology domain is an important issue for mobile software architects. These patterns could be seen as forming a catalogue of core patterns for the mobile domain, by analogy with the core J2EE patterns mentioned above.

In this chapter, we present a study of patterns for the mobile domain and an approach we have developed for organizing and identifying suitable collection of patterns.