Specifying Patterns for Mobile Application Domain Using General Architectural Components

Oleksiy Mazhelis[1], Jouni Markkula[1], and Markus Jakobsson[2]

[1] University of Jyväskylä,
Information Technology Research Institute,
P.O. Box35, FIN-40014 University of Jyväskylä, Finland
{mazhelis, markkula}@titu.jyu.fi
[2] Ab SESCA Technologies Oy,
P.O. Box77, FIN-68601, Jakobstad, Finland
markus.jakobsson@sesca.com

Abstract. Software companies adopt patterns as a means to improve architecture and design practices. During recent years, the application of patterns has extended from general software applications to specific problem domains. In a new domain, suitable patterns fitting to the essential design problems in the new context need to be identified. In this paper, we introduce a general architectural model of mobile applications, which can be used to identify and organise essential patterns in mobile-application design process. This model is employed to construct a high-level architecture of a particular application. For each component of the architecture, the model may suggest candidate patterns that can be used for elaborating the component. Subsequently, the results of the design process are used iteratively to further develop the architectural model. The presented model is verified and tested by employing it to address the design problem of supporting multiple user interfaces in a real mobile application product.

1 Introduction

Mobile technology domain is a challenging environment for software design. The complexity and peculiarities of the environment as well as fast development of the technologies beget design problems which are not present in traditional application environments. This, in turn, often necessitates design solutions differing 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 quickly evolving technology, and they need to be implemented and ported quickly to new technologies and platforms. Thus, mobile applications need to possess specific qualities including, in addition to required functionality, the qualities of modifiability, reliability, efficiency, etc. These qualities are largely achieved by architectural means [1], and therefore, careful architectural design is important.

Patterns have been introduced and adopted by the software community to support systematic design of architectural solutions with the desired qualities. A number of

F. Bomarius and S. Komi-Sirviö (Eds.): PROFES 2005, LNCS 3547, pp. 157 – 172, 2005.

design patterns and pattern systems have been elicited during last decade. A pattern in software engineering has been defined by [2] as a description of communicating objects and classes that are customized to solve a general design problem in a particular context. Patterns should present a solution to a problem in a context, and their essential elements include [2]: 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 [3]. They can be divided into general (or domain-independent) patterns [2, 3] and domain-oriented patterns tailored to specific areas of applications and/or to specific development tools or platforms [4, 5].

Patterns are typically collected in pattern catalogues. The patterns in catalogues are organised according to certain characteristics, and relationships between them are described. To be useful, the catalogues should contain only a *restricted* number of patterns which reflect the most commonly encountered design problems, in general or in a specific domain[1]. Among domain-independent pattern catalogues, one of most well-known is the famous GoF catalogue presented in [2]. An example of a domain-specific catalogue is the Core J2EE Pattern Catalog[2] [4], where many of the patterns useful in designing enterprise applications with J2EE are collected.

As noted above, the mobile environment represents a technology domain which has its peculiarities strongly affecting the design solutions. In solving a design problem with patterns, these peculiarities should be taken into account in order to identify suitable patterns. Namely, the patterns fitting to the current design problem and *matching the context* of current application need to be selected [2]. While rich collections of patterns are already available through different sources, their applicability in mobile domain needs to be evaluated by the designer, as the consequences of patterns (and possibly, other characteristics as well) may have different significance in the context of mobile applications. Some domain-independent and domain-specific patterns available in pattern catalogues might appear inappropriate in this context or require modification. On the other hand, some patterns address the problems often encountered in designing mobile applications, and, hence, are highly valuable in the mobile domain.

The patterns applicable in 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 [6] as well as the Symbian two-phase construction and the cleanup stack idioms [7]. The patterns from other domains *directly applicable* in the mobile domain can be exemplified by Model-View-Controller (MVC) [7] or by the Remote façade pattern [8, 5].

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 mobile technology domain is an important issue for mobile software architects. These patterns could be seen as forming a catalogue of

[1] By domain, we refer to a particular set of constraints applied to the platforms, problems, etc.
[2] Available e.g. at http://corej2eepatterns.com/Patterns2ndEd/index.htm