Patterns for Mobile Applications

Markus Aleksy
University of Mannheim, Germany

Martin Schader
University of Mannheim, Germany

INTRODUCTION

The increasing adoption of mobile terminals, together with the progressive development of handheld devices and the simultaneous improvement in the infrastructure of wireless communication plays a more and more important role in the development of new kinds of mobile applications. Applications that are executed on mobile terminals often require both a simultaneous interaction with other users of mobile terminals as well as with fixed or location-dependent services. Although this development is very favorable from the view of the end user, it confronts application developers with the problem that they must manage the growing complexity of the mobile applications.

A main characteristic of mobile applications is their dynamics. It is therefore important that during development of a mobile application, its dynamic adaptability—as one of the most important design criteria—must be taken into account.

In this article, we present a pattern-based structure for the development of mobile applications. After a short description of the concept of patterns, we demonstrate how these may be used in the context of the development of mobile applications.

PATTERNS

Patterns are simple and concise solutions for programming tasks frequently appearing in practice. The architect Christopher Alexander (Alexander et al., 1977; Alexander, 1979) is regarded as being the intellectual father of the patterns movement. The structure of a pattern follows certain rules and is based on the elements listed below (Gamma, Helm, Johnson, & Vlissides, 1995):

- **Pattern Name:** The name of a pattern is usually short yet descriptive and acts as an addition to the design vocabulary.
- **Problem:** Patterns should include a short description of the problem they intend to solve.
- **Solution:** The solution to the problem is described in a generally applicable way. The elements of the solution are described along with their relationships and responsibilities.
- **Consequences:** While the main consequence of using the pattern is solving the problem, there are often side effects. In order to make it easier to understand the trade-off involved in using the pattern, it is important that the potential drawbacks are explained.

Of course, this is not the only possible structure for describing a pattern. Gamma et al. (1995) also provide an extended schema, and other authors offer alternative descriptions as well (e.g., Fowler, 1997).

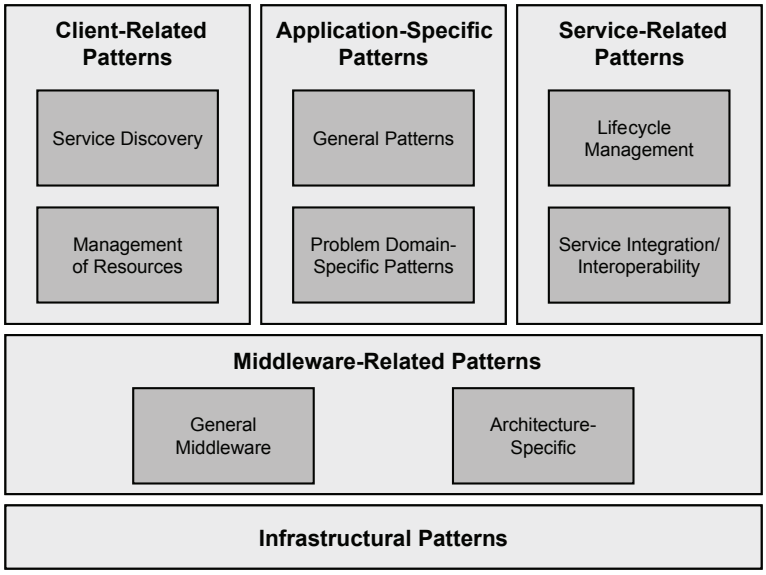
A pattern documents a comprehensive and good solution won from experience to a problem frequently recurring during program design. The engineer-like approach at the design of object-oriented software consists of identifying one or more patterns for a concrete programming task, which helps in coping with this task. The documentation of the patterns supports the development process, from the complete specification of the set task up to its implementation in a concrete programming language; in addition, it supplies a common language frame for the discussion of the envisaged solution between the developers.

The primary benefit of a pattern lies in the description of a solution for a certain class of problems. Additional advantage arises from the fact that every pattern has a name. This simplifies the communication among software developers since one may then discuss a software structure on an abstract level. Thus, patterns are first of all programming language independent. During the design of object-oriented software, they serve as a recognized aid to structuring the outline. A detailed introduction to the concept of patterns can be found among others in Gamma et al. (1995) or Buschmann, Meunier, Rohnert, Sommerlad, and Stal (1996).

PATTERNS FOR THE DEVELOPMENT OF MOBILE APPLICATIONS

When developing a mobile application, a programmer can meanwhile resort to a variety of different patterns. One can distinguish among patterns with a different coverage (see Figure 1):

Figure 1. Patterns for the development of mobile applications



- infrastructural patterns,
- middleware-related patterns,
- client-related patterns,
- service-related patterns, and
- application-specific patterns.

Infrastructural Patterns

Infrastructural patterns describe solutions for the administration of mobility (mobility management functions). They illustrate which elements are needed for the realization of architecture. Andrade, Logrippo, Bottomley, and Coram (2001) identify two types of patterns that fall into this category: *architectural elements* and *functional behaviors*. The patterns *home and visitor databases* as well as *security database* fall into the first category. The first pattern solves the problem of mobility between different local areas (*roaming*) of the same supplier or of different suppliers. The second is aimed at the realization of security and privacy functionality. Besides that, the authors identify broader functional behavioral patterns, such as *temporary identification, paging, authentification, ciphering,* as well as *location registration.*

Middleware Patterns

Building on the infrastructural patterns introduced above, a variety of middleware patterns can be used. Here, middleware functions as the link between different spatially distributed software components. We introduce different middleware-related patterns in the following. These were developed mainly for the realization of non-mobile applications, but it is possible to transfer them into the domain of mobile computing as well.

Patterns that fall into this category can be classified as *general design patterns,* which can be used for the realization of (almost) all types of distributed architectures, or as *architecture-specific patterns.*

General patterns, which can be used for the realization of a basic infrastructure for object-oriented systems, are described for example by Buschmann et al. (1996), Schmidt, Buschmann, Stal, Rohnert, and Sommerlad (2001), and Völter, Kircher, and Zdun (2002).

Patterns such as the *forwarder-receiver* or the *publisher-subscriber* (Buschmann et al., 1996) may be applied in a variety of applications and architectures.

On the other side, middleware patterns exist which are specific for certain architectures. For example, during the realization of a mobile client-server architecture, the patterns introduced by Völter (2001) can be used on the server side. Aarsten, Brugali, Menga, Brown, and Hirschfeld (2005) present patterns particularly suitable for the realization of three-tier client-server architectures. Another pattern falling into this category is the *broker* pattern (Buschmann et al., 1996), which gained wide popularity in the context of the common object request broker architecture standard (CORBA) defined by the Object Management Group (2004).

Patterns for the Development of Mobile Clients

During development of a mobile application, dealing efficiently with the resources of the mobile client represents