A Software Engineering Model for Mobile App Development

As we mentioned early in the book (see Chapter 1), to successfully develop a mobile software solution you should follow an engineering process that helps you address the specific characteristics of mobile software. We refer to this as mobile software engineering. Such a process is ideally highly agile, incorporating several macro and micro-iterations. Of course, you can follow any process you prefer, or develop your software without following any process at all. There is a range, from a strict waterfall model to cowboy coding. As a rule of thumb the more complex a project is and the more coordination it requires the more formalisation in terms of processes is advisable. Experience has shown that so called agile processes are very appropriate with software for fast-paced markets, which is definitely the case in mobile app development.

From a high-level point of view, there is no difference between developing software for desktops, servers, the Web or for mobile devices. It is all software engineering and the basic steps are always the same: requirements, design, programming, testing, and deployment. However, it is the detail that makes the difference. From experience we can say that it is not possible to simply transfer the techniques of traditional software engineering one-to-one to mobile software engineering without significant modifications.

As we are talking about the bada platform for mobile applications in this book, our prime interest in this appendix, naturally, is the mobile software engineering process. In the following pages we will introduce one such agile mobile software engineering process that subsumes several recommended best-practices. Of course, you are free to choose whichever engineering process you want to follow.
and whether to comply with all the phases we introduce, pick out the most relevant for your needs, or not to use it at all. We will also show where and how the bada platform and its tools support the described phases and best-practices.

### Some Mobile Software Engineering Best-practices

For developing mobile software, we recommend deploying ideas from agile development initiatives (stated in the agile manifesto\(^1\)) such as adaptability, iterations, and making heavy use of prototyping and diversified testing as early as possible in the process.

Prototypes can be exploited in nearly any phase of the engineering of a mobile software solution. They are primarily helpful in eliciting requirements or to get a common understanding with various stakeholders early in the project. Testing surely is not unique to mobile software engineering but must be treated and executed differently, as we will argue throughout this appendix.

The mobile software engineering process that we introduce here is subdivided into three major phases:

1. Feasibility and economic efficiency analysis phase
2. Software product realisation phase
3. Distribution phase

The following sections discuss each of these phases in detail and Figure C.1 gives an illustration of the whole process.

---

\(^1\)See [http://www.agilemanifesto.org/](http://www.agilemanifesto.org/).
Phase 1: Feasibility and Economic Efficiency Analysis

This first phase should help all the stakeholders to get a better picture about the feasibility, the acceptance, and the potential economic success of a mobile software solution. By economic efficiency analysis we mean that very early in the project lifetime it can be determined whether it is worth pursuing a mobile software development project. This first phase is composed of four stages: requirements engineering, design drafting, early prototyping, user acceptance testing, and a milestone at the end of the phase.

Stage 1.1: Requirements Engineering

The first step in this analysis phase is the elicitation and understanding of the requirements. The business requirements cover considerations about the business model and distribution policy. This is input that usually originates from the product management process in which our mobile software engineering process is embedded.

User requirements are the starting point and represent what users need and what they expect from mobile software. These requirements must be collected and can be maintained in the form of use-cases or user stories. Early prototypes can be used for this, because users can get a more tangible impression of the future system. Early prototypes can be low-fidelity ones, such as drawings on paper. More advanced ones can be created by using electronic images that can actually allow a sequence and simple interactions to be shown. Even more advanced is the use of simple user interfaces (UIs) that can be created fairly easily and quickly by using the UI Builder provided by the bada IDE. The application envisaged can be sketched and simple user interaction (clicking on buttons or exploiting multi-touch) can be modelled and even deployed and demonstrated on a real hardware device. This helps the user even more to think of the future use of an application, which is directly reflected by the quality of user feedback.

For mobile applications we recommend focusing on addressing a clearly defined problem or need with as little functionality as required. Many applications seem overloaded and, as a result, are difficult to use. Focus on the core and address rather fewer requirements. We are not talking about desktop applications. We have already outlined the differences between mobile and desktop apps. The cognitive resources and the attention timespan is substantially more limited. Traditional software engineers tend to underestimate the impact that a mobile setting will have on application usage patterns.
Stage 1.2: Design Drafting

The drafting of the design deals with two aspects. The first aspect is the draft of the dialog flow logic and the UI design. This is the only part of the software that is exposed to users and, hence, the only part they can experience. Experience has shown that it is essential to provide a simple yet attractive UI. If an application is too complex to understand within the first minute, or it is ugly looking, the user will most probably not launch it a second time.

The second aspect deals primarily with software component architecture considerations. The UI design activities are conducted in various micro-iterations with feedback from stakeholders.

Stage 1.3: Early Prototyping

Early prototypes embody the requirements gathered and the design developed so far. They are a good means to get a common understanding. Some rudimentary functionality should be in place such that minimal interaction is possible. That can be achieved by paper prototypes, an interactive slideshow or a clickable UI mock-up. The goal is to find a good balance between a fast and cost-effective prototype and providing a user experience that comes close to the final product.

As mentioned above, the bada tools give ideal support for building good prototypes quickly. A further advantage is that these prototypes can be reused for the later programming as they are, without the need of redoing all the work for the UI. This is a result of the clear separation between UI design and programming brought to you by bada.

Stage 1.4: User Acceptance Testing

This is an optional stage. However, we recommend planning and executing tests with users that are not involved in the project. These tests can be short in length but should be as close to the real-world context as possible. This way, many problems that would go unnoticed in a lab environment will become apparent, such as bad readability under sunlight or overly complex and confusing user interfaces. These kinds of trials will almost always yield highly valuable feedback about the future acceptance by users and thus potential success in the market. After a test session there may be one or more iterations back to the requirements engineering stage depending on available time, and quality and cost targets.

The first phase ends with the milestone Decision for Continuation, which entails the decision to pursue or abandon the software project. With the knowledge gained in this phase – particularly arising from the early acceptance tests – the potential success of a mobile solution can be estimated much more clearly, which helps to assess the economic efficiency.
Phase 2: Software Product Realisation

The second main phase deals with realising the software solution. It builds on the work in the first phase, and as many of the results as possible should be reused. In following the agile engineering process, this phase is also characterised by many iterations, incremental development (‘first things first’), and a high degree of internal and external communication.

The phase is composed of six stages: requirements reviewing, design detailing, defining test cases, programming, testing, user acceptance testing, and a milestone at the end of the phase. Again, we would like to emphasise that we present here a complete mobile software engineering process. With reference to our advice about working in an agile way, which implies adaptability, you are free to choose whichever stages you want to deploy in your software project.

Stage 2.1: Requirements Reviewing

Prior to starting with the detailed software design, programming, and testing, the requirements should be reviewed and revised – ideally together with all stakeholders.

Stage 2.2: Design Detailing

In this step, the available UI and architecture designs of the first phase are detailed into much more fine-grained levels – down to element and component level. Although, mobile software engineering is a comparatively young discipline, some (mostly vendor-driven) initiatives have emerged that provide guidelines for UI design. Although you are not bound to use these, we strongly recommend complying with these guidelines because it simply increases the usability of your product and its acceptance.

Also the bada platform has introduced such guidelines. These are implicitly used when you use the open application programming interfaces (APIs) for the UI and extended UI controls. Apart from that, the IDE ships with a help document that gives further information about best-practices for ideal UI design.

Stage 2.3: Defining Test Cases

Testing is absolutely necessary in any software engineering endeavour. In mobile software engineering it is more multifaceted and more variables need to be taken into consideration. The definition of test cases is the first activity related to testing. The cases can be derived from the requirements.
**Stage 2.4: Programming**

During this stage the designs should be transformed into program code that can finally successfully pass all test cases. Although mobile devices are becoming more and more powerful, it is nevertheless highly recommended to write efficient code with an eye on conservative computation to conserve as much battery capacity as possible. Stretching availability of battery power is crucial, as we argued in Chapter 1.

The bada IDE ideally supports you with all you need for writing your program code, such as syntax highlighting, code completion, comprehensive documentation, and example material, and all the necessary compiling, linking, deploying, and running functionalities.

It is also important to understand that every mobile application is subject to a ‘technical’ application lifecycle, which subsumes initialising, running, terminating, and terminated. The bada platform and the IDE that you use to access it provide support for dealing with this app lifecycle and doing the right things in the appropriate stages (through well-defined methods).

**Stage 2.5: Testing**

In the mobile software engineering process, much emphasis must be placed on the testing stage. It is important to differentiate between the testing platform and the real platform. The testing platform is usually a desktop computer that runs an emulation of the mobile device. Unfortunately, emulators exhibit great discrepancies with real mobile devices. Emulator tests are in ideal lab environments where context factors such as position or light conditions are either not considered or simulated. Hence, it is necessary to test software on the real target mobile device and in the real-world environment where the software is intended to be used. All the ‘controllable’ variables must be caught by the software, such as behaviour according to device capabilities, adapting to screen size, masking network disconnections, or loss of GPS signal.

These last three stages (defining test cases, programming, and testing) are characterised by many micro-iterations and in reality are partly also executed in parallel. In particular, testing naturally iterates with the test case definition and programming stages. It usually also iterates with the design detailing and the requirements reviewing stages.

**Stage 2.6: User Acceptance Testing**

User acceptance tests at the end of the second phase are again optional but recommended. This way, engineers can make sure to really meet users’
requirements with a software version that increasingly resembles the final state. This type of testing can be repeated and the outcomes can be fed back into earlier steps of this phase. Human–computer interaction techniques such as audio/video recordings, questionnaires, cooperative evaluations, focus groups, or controlled experiments can be deployed and are beneficial.

After stakeholders accept the existing version of the mobile software, it can be denoted as a release. This means that the milestone Version Released is fulfilled, and the next iteration of the same phase (to realise a further increment) or the successive phase can start.

**Phase 3: Distribution**

This phase mainly deals with bringing the mobile software product into the market to the users. This phase is less iterative than the other two. It is composed of the three stages: marketing, preparing for deployment, product maintenance, and a milestone at the end of the phase.

**Stage 3.1: Marketing**

The marketing stage exemplifies where and how the processes of mobile software engineering and product management overlap and complement one another. This stage serves for the finalisation of the business model and distribution policy, and for preparation of the actual market entry of the software product. Marketing is partly in parallel with other stages, and starts during earlier phases.

**Stage 3.2: Preparing for Deployment**

In this step, the software must be prepared (i.e. certified) such that the required functionality of a mobile device can be accessed appropriately. It must be decided which components and content are packaged together and which parts may be downloaded or installed on-demand. Also the distribution channel must be prepared, which usually involves requesting app store accounts and becoming familiar with the relevant policies. Finally, the mobile software is physically distributed and installed on the end-users’ devices.

**Stage 3.3: Product Maintenance**

Maintenance covers support, bug fixing, and feedback integration. This step – depending on the significance of the reported issue and the policy of the
application provider – can lead to an additional iteration with the preparing for deployment stage. Also, the level of maintenance and the commitment to it depends on the provider and can range from 24/7 service to no support at all.

The mobile software engineering process ends with the end event (milestone: Project End) that was defined by the stakeholders and the project manager.