Modeling Android applications using UML

Abilio G. Parada, Thiago A. Alves, Lisane Brisolara

{agparada,taalves,lisane}@inf.ufpel.edu.br

Universidade Federal de Pelotas (UFPel)
Pelotas – RS – Brasil

Abstract

The applications development for Android platform differs from traditional software development. These applications are usually complex, due to the characteristics imposed by the platform, thereby increasing the design complexity. Moreover, the designer has to worry about a reduced time to market. This way, approaches to accelerate the development and deal with complex systems are required. UML models are used in whole conventional software design process to provide abstraction. Allied to Model-driven Engineering (MDE), it provides also automation, through model transformations. Thus, this paper presents a UML-based modeling approach for Android applications, which allows designer have benefits from usage of UML and represents the first step to apply MDE on its development.

1. Introduction

The development of Android [1] applications differs from that used for traditional applications, because it requires the use of new APIs and new concepts like Activity and Service specially defined for mobile applications, in addition, it obliges designer to consider the Operating System and hardware characteristics. Furthermore, these applications are linked a competitive market, which has motivated the research of approaches able to accelerate the production process, and maintain the quality with low cost, thus making the competitive product. Frequently models are used to handle high complexities [2], because models have few details and are easier to build, facilitating the understanding and detection of problems.

The Unified Modeling Language (UML) [3] has become the standard modeling language for object oriented software, providing insight into behavioral and structural design. The *Model-driven Engineering* (MDE) [4] models are used in all phases of software engineering (analysis, design, implementation, and testing), not only in the documentation phase. Models are considered as primitive artifacts, which evolved and are transformed until to be possible to automatically obtain an implementation from it. The UML combined with MDE can provide abstraction and automation, accelerating the software development process. According to a study in the Motorala, MDE may reduce the time spent on software production by 70% [5].

Recently, Model-driven Engineering (MDE) approaches [6], used by the software community, have gained attention for the embedded community, promising automation and abstraction for embedded software development. To support it, code generation approaches should be defined and tools must be available.

Some authors have defended the use of MDE approaches for mobile devices development as in Wang [7], where the main mobile components are represented using UML models. However, mobiles applications can require different modeling resources. To support Android applications development in accordance with the MDE principles, this paper presents a model which represents the main components used in the development of an Android application.

This paper is organized as following. Section 2 presents the Android and discusses the development for this platform. The proposed model is presented and discussed in Section 3 and Section 4 discusses related works. Section 5 presents conclusions and future works.

2. Developing to Android

The Android is an operating system based on the Linux *kernel* held by Google to mobile devices. This uses Java as native language, however Android applications are executed on the Dalvik [8] virtual machine, and not on the traditional Java Virtual Machine (JVM).

Among the components used to describe an Android application, *activity* and *service* should be highlighted. The *activity* usually corresponds to display screens, and *service* is responsible for performing a background task. These components typically are the most part of the code description, and also increasing the project complexity. Both, *activity* and *service* have methods with generic components and are usually extended by the designer.

The Android platform also provides resources for communication between functionality and data. The communication between functionalities is given by *Receiver of Broadcast and Intents* (RBI). For this communication, the object *Intent* must be passed as a parameter for the RBI to search the functionality.

However, a data requesting is treated by the *Content Provider* (PC). The requesting is indicated through URI (*Uniform Resource Identifier*), which provides the standard access to PC.

3. Modeling Android Applications

As discussed before, for completely modeling Android applications, specific concepts used to design the application should be adequately represented. This section presents the model developed to support the UML-based modeling for Android applications. The class diagrams are use to describe structural components, representing the components Activity and Service, classes, its attributes, and methods. To describe the behavioral components are used sequence diagrams, which represents RBI, PC and methods behavior.

The modeling of main structural components and behavioral components are presented in subsection 3.1 and subsection 3.2, respectively.

3.1. Structural Components

Fig. 1 illustrates the structural overview for an Android application, represented by a class diagram. The classes *Activity* and *Service* are in gray, representing the Android structural components, and in white, the components defined by the designer, Main and Background. The diagram allows represent the inheritance relationship between these elements.

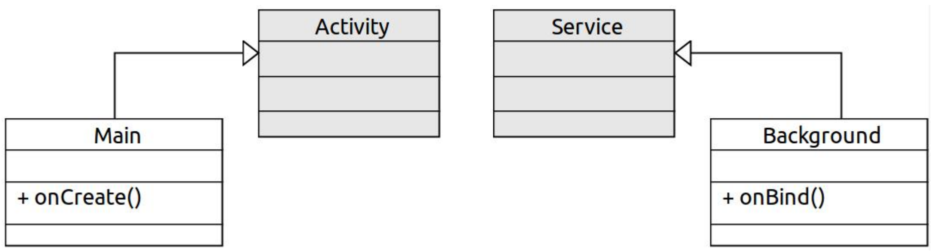

Fig. 1 – Class Diagram – Overview of the structural components.

The classes *Activity* and *Service* have several methods that must be extended by the designer, as *onCreate*, *onStart*, *onDestroy*, *onBind* and others. Thus, the identification of these methods in the model, able the generation of standard code characteristics of each method. As the goal of this paper is the modeling of the main components used by an Android application, the remaining structural components of the system were omitted here. These components must be defined by the designer, exploring the main resources provided by UML class diagrams, as the class definition, interfaces, attributes, operations and relationship.

3.2. Behavioral Components

The system behavior view is represented for sequence diagrams, which represents iteration, conditional, and message exchanges. Fig. 2 illustrates a data request to Content Provider (PC) from the class *Main*. In this figure, the object *Main* requests the *contact* data to object *PC*, through message change. The behavioral description of shared data, provided in Android by PC, is illustrated in this sequence diagram. The use of sequence diagram for modeling this data requesting, can provide to the designer a global data request overview, of each component, allowing a better redefinition of requests.

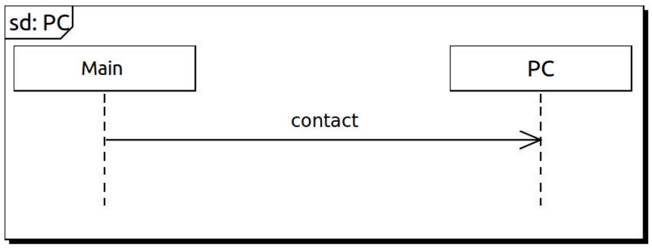

Fig. 2 – Sequence Diagram – Description a data request to the *Content Provider*

The sequence of activities necessary to request an action to the Broadcast and Intents Receiver (RBI), by a component is illustrated in Fig. 3. The object *Main* represents that this activity will be held in the class *Main*, first is made the request of the *Intent*, through a sent message called *Intent*, like *ACTION_CALL*, to the IPC. The return of this operation will be assigned to the object *Intent*. The second action is a functionality request which intends to RBI, performed by the method invocation *startService*, and passing the object *Intent* as argument.

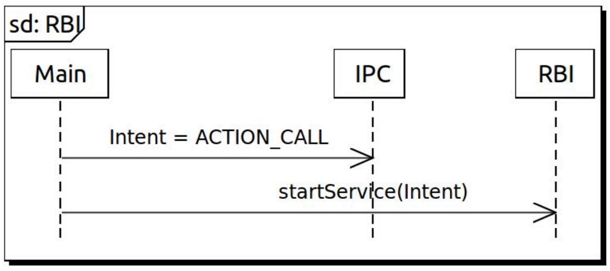

Fig. 3 – Sequence Diagram –Description a request to the Broadcast and Intents Receiver.

The methods inherited from *activity* and *service* must be implemented by the programmer. The *onCreate* method, inherited from the *activity*, describes the first activities executed by an application, when it goes on operation. This method uses the PC and RBI described in the Fig. 2 and Fig. 3, respectively. Fig. 4 illustrates the basic behavior of *onCreate* method, where can be observed the references for PC and RBI sequence diagrams, indicated by the *Ref* fragment. After that, the requested to RBI is executed by sending the message *onStart* to start the service, and *onBind* to make a persistent connection to a service.

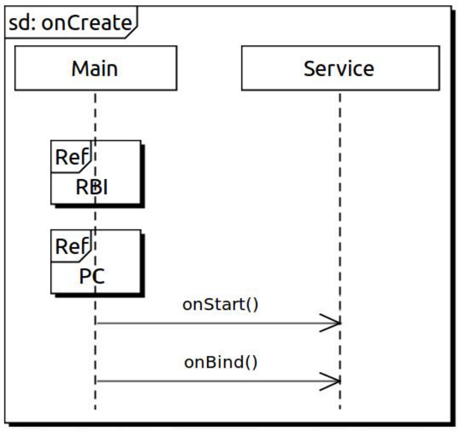

Fig. 4 – Sequence Diagram – Behavioral description of the method *onCreate*.

4. Related works

For modeling traditional software using UML, a huge number of tools are available and some of these tools are able to generate code from UML models. Among these tools, we can highlight Rational Rose [9], as a commercial tool, and Eclipse Model Tool [10] provided by Eclipse group. Beside of these, GenCode [11] is an academic effort to provide a tool able to generate Java code from sequence and class diagrams. However, these tools do not consider the specific aspects required for the Android application development.

Recently, AndroMate [12], an Eclipse plug-in, was proposed to support the modeling and code generation for Android platforms. The modeling in this tool is made using model elements symbolized as boxes which represent generic functions. The main elements are *ActivityContainer*, which is responsible for representing the active applications, as for example, the selection of a contact from a contact address application, *DataTypeContainer*, which represents the applications data type, and *ViewContainer*, corresponding to the screen layout, like linear layout or scroll view. However, this tool does not use the UML standard language, thus the designer does not have whole benefices of UML usage, as use of a standard language and its modeling resources (e.g. class, and relationships. Our work is an initial effort to propose a MDE approach suitable for Android applications, which will use UML modeling resources and support automatic code generation.

5. Conclusions and future work

This paper presents a modeling approach for the Android applications, focusing on the use of UML modeling resources to completely model concepts evolved in the design of Android applications like Activity, Service, Content Provider, and others relevant concepts. This approach can provide a behavioral and structural view of the system, using UML class and sequence diagrams.

As future work, a case study should be conducted which consists on the modeling of an Android application using UML and the modeling approach proposed in this work. However, for the effective use of MDE, tools should be able to apply the required model transformations in order to obtain the final implementation from them. As our group has previously developed GenCode, a tool which generates Java code from sequence and class diagrams, as future work, we plan to extend this tool in order to support the code generation for Android applications.

6. References

[1] Android (2012), Anatomy of an android application. Avaliable in: <http://code.google.com/android/intro/anatomy.html>.

[2] Selic, B. (2003). Models, software models, and UML. UML for real: Design of embedded realtime systems (pp. 1-16). Boston: Kluwer Academic Publishers.

[3] OMG. Unified Modeling Language (UML) (2012). Avaliable in: <http://www.omg.com/>.

[4] Selic, B. (2006). UML 2: A model-driven development tool. Model-Driven Software Development. IBM Systems Journal, Riverton, v. 45, n. 3, p. 607-620.

[5] Weigert, T. (2011). Practical Experiences in Using Model-Driven Engineering to Develop Trustworthy Computing Systems. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing -Vol 1 (SUTC'06).

[6] Espinoza, H.; Cancila, D.; Selic, B.; Gérard, S. (2009) Challenges in Combining SysML and MARTE for Model-Based Design of Embedded Systems. In: Proc. of the 5th European Conference on Model Driven Architecture - Foundations and Applications.

[7] Wang, Z. (2011). The study of smart phone development based on UML. Computer Science and Service System (CSSS). P 2791 – 2794.

[8] Dalvik (2012), Dalvik Virtual Machine. Available in: <http://www.dalvikvm.com/>.

[9] IBM (2011), IBM Rational software. Available in: <http://http://www.ibm.com/software/rational/>.

[10] MDT (2011), Eclipse Model Development Tools. Available in: <http://www.eclipse.org/modeling/mdt/>.

[11] Parada, A.; Siegert, E.; Brisolara, L. (2011). Generating Java code from UML Class and Sequence Diagrams. In: Workshop de Sistemas Embarcados, 2011, Florianópolis. I Simpósio Brasileiro de Engenharia de Sistemas Computacionais Vol. 1, 2011.

[12] AndroMate (2012), The Android Modeller and Code Generator. Available in: < http://www.lab.telin.nl/~msteen/andromate/>.