MDD for Smartphone Application with Smartphone Feature Specific Model and GUI Builder

Koji Matsui and Saeko Matsuura
Division of Electrical Engineering and Computer Science,
Graduate School of Engineering and Science.
{ma14097@, matsuura@se.}shibaura-it.ac.jp

Abstract—**Unlike general PC applications, smartphone applications have three innovative features that make useful mobile services a possibility. Conventional code-centric development tools used for general PC applications are not efficient for developing high-quality software with mobile features. The difficulty with conventional development is because of the variety of platforms and operations. Model Driven Development (MDD) is a promising approach to develop high-quality software products efficiently. To develop richer applications using such features, we propose a UML-based MDD method. This method uses a Smartphone Feature Specific Model and a GUI builder, independent of any specific OSs.**

Keywords-MDD; UML; Smartphone Application; GUI builder.

I. INTRODUCTION

Smartphone applications have three innovative features that present a possibility of useful mobile services. The first feature is that the device is equipped with various types of hardware. This enables the user to input a variety of data; for example, user actions that cannot be expressed by characters. The second feature is that the application can be easily extended by connecting external applications, such as *Intent* in *Android,* or *URLScheme* in *iOS* using various communication mechanisms. The third feature is a set of rich User Interface components for multi-touch devices that enables us to use the interface to improve the operability of a smartphone.

The development of applications for a smartphone is a complicated task because of the variability of platforms and the number of different devices that need to be supported. Moreover, the basic design of a target application that includes UI operability needs to be analyzed at the early stages of development to reduce the need to rework.

GUI builder allows a developer to arrange widgets using a drag-and-drop WYSIWYG editor, so that he/she can develop the user interface of the application in an intuitive manner. However, the intuitiveness of the interface is entirely dependent on the specific programming language and the analysis of the application logic. This relationship tends to be insufficient in regards to the first two features.

We propose a unified modeling language (UML)[1]-based Model Driven Development (MDD) method using a smartphone feature-specific model and a GUI builder that is platform independent.

The remainder of the paper is organized as follows. Section II discusses how to develop smartphone applications efficiently. Section III explains how to model the smartphone application using suitable development tools stated in our approach. Then, the related work is discussed in Section IV.

II. PROBLEMS IN DEVELOPMENT OF SMARTPHONE APPLICATION

Since mobile services with the abovementioned features support varied platforms and operations, it is difficult to implement conventional code-centric development to develop such a system efficiently. MDD [2] [3] is a promising approach to develop high-quality software products efficiently because it enables code generation and has high traceability.

The issue with changeability of platforms can be solved by separating concerns about platforms. The Platform Independent Model (PIM) and the Platform Specific Model (PSM) use UML. However, to realize appropriate operability, we need to design a system that uses a concrete screen image.

A developer can use GUI builder [4] for the specified OSs and develop application user interfaces in an intuitive manner. However, the intuitiveness of the interface is entirely dependent on the specific programming language and the analysis of the application logic. This relationship tends to be insufficient. Thus, the product developed using GUI builder is difficult to reuse in other applications and cannot follow various requirements changes.

UML is a well-known general-purpose modeling language that provides a standard method to visualize the design of a system. There are several convenient UML editors, such as astah* [5]. astah* and other UML editors are effective tools to design the static structure and behavior of a system; however, these tools are unsuitable to design GUI in an intuitive manner.

The problem is how to efficiently develop smartphone applications that deliver feasible static content, as well as an intuitive behavioral model. Further, these applications must also be independent of any specific OSs.

III. UML-BASED MODEL DRIVEN DEVELOPMENT METHOD

A. Overview of Development Process

To solve the above-mentioned problem, we propose a UML-based MDD method using a *Smartphone Feature Specific Model* and an original *GUI builder* independent of any specific OSs. Figure 1 shows an overview of our method.

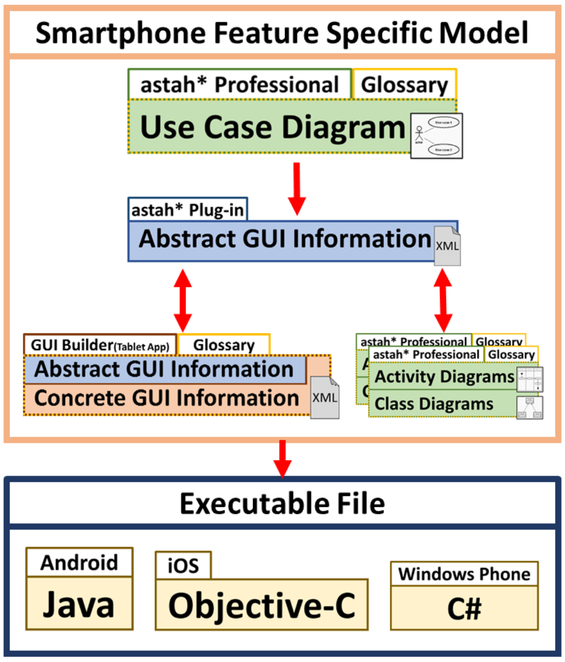

Figure 1. Overview of our Method

Use case analysis [6] is known as an effective method to define functional requirements. Therefore, because a use case represents a basic unit of function that is used by an end-user, we begin the method by constructing a use case diagram

The Smartphone Feature Specific Model consists of two types of data edited by such different design views as a UML modeling tool and the GUI builder.

The first data is a UML Model specified by a glossary of smartphone features, as shown in Figure 2. A UML Model consists of a use case diagram and a pair of an activity and a class diagram. An activity diagram and a class diagram correspond to a use case. A developer edits the pair using the UML modeling tool, astah*.

The second data is defined by the GUI builder and consists of *Abstract GUI Information* and *Concrete GUI Information.* The former consists of abstract components that are common in the Android, iOS, and Windows Phone SDK [7] [8] [9]. Moreover, the second data is connected with the UML Model by a mapping rule based on a meaning of a use case. The latter shows properties such as size, position, font, color, and concrete values, which are added to the first data.

The mapping rule defines mutual transformation between both data defined by the UML modeling tool and the GUI builder. The data of the UML Model is extracted using the astah* API Plug-in.

After a developer edits a target application using proper views that he/she thinks fit to design such aspects as function, structure, behavior, and operability. *Smartphone Feature Specific Model* data is written in XML and can be translated into codes in specific programming language such as Java, Objective-C, and C#.

B. Glossary of smartphone features.

Figure 2 shows a glossary of smartphone features mentioned in Section I. Smartphone features are classified into four classes: *View*, *Gesture*, *State*, and *ExternalSystem*. These classes are used as basic components of the smartphone specific model.

View consists of 13 *Widgets* and 5 *Layout* classes that are used for editing on the GUI builder. *Widgets* are

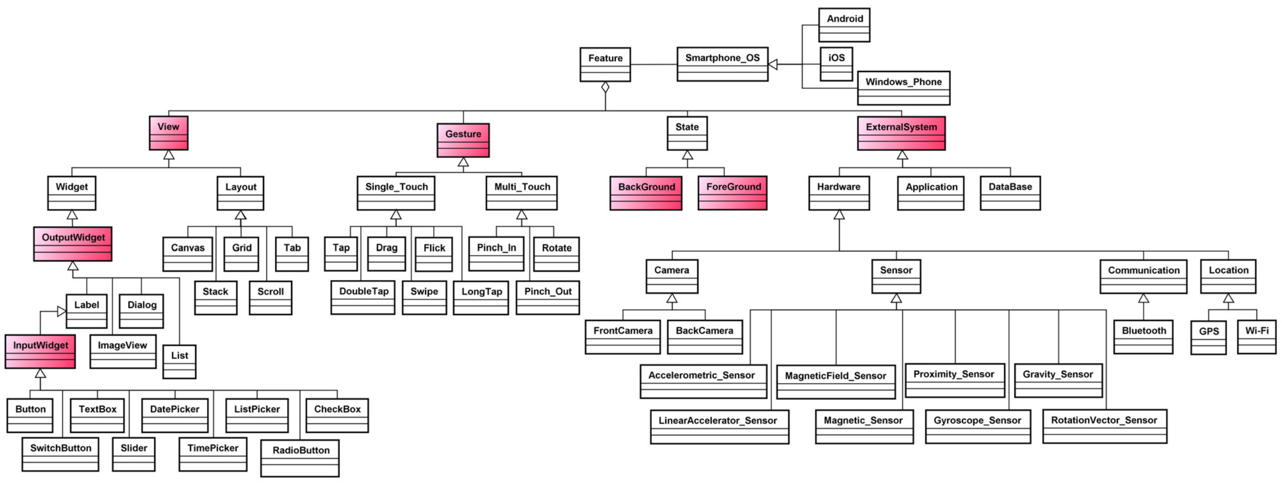

Figure 2. The Glossary of Smartphone Features

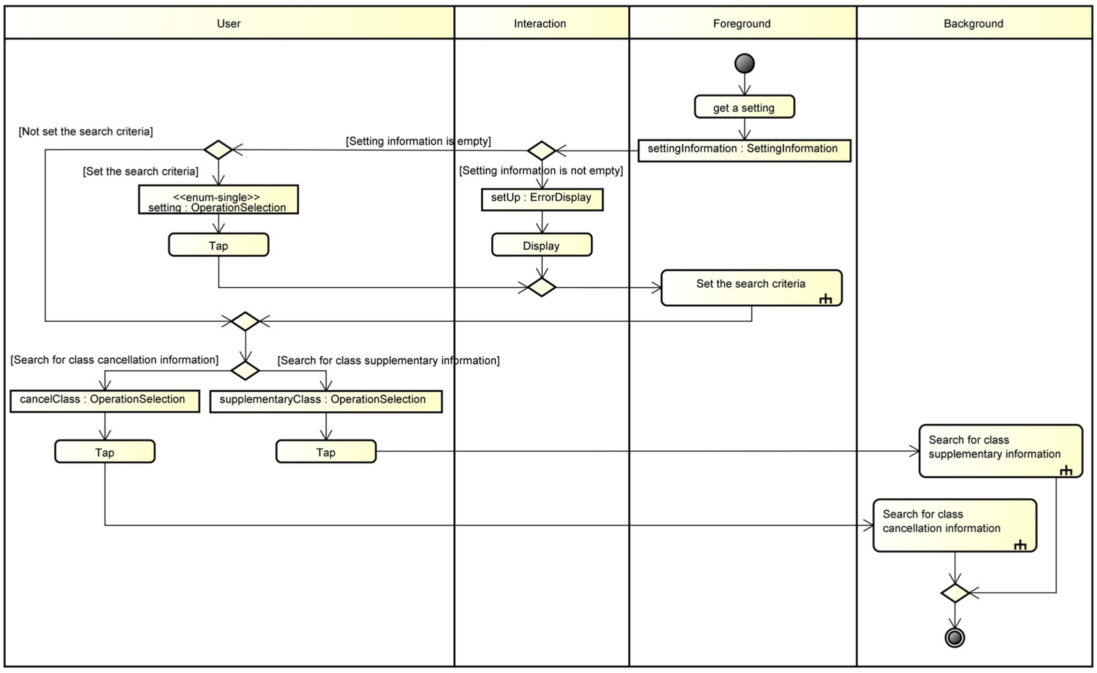
User
Interaction
Foreground
Background
get a setting
settingInformation : SettingInformation
[Not set the search criteria]
[Setting information is empty]
[Set the search criteria]
[Setting information is not empty]
<<enum-single>> setting : OperationSelection
setUp : ErrorDisplay
Tap
Display
Set the search criteria
[Search for class cancellation information]
[Search for class supplementary information]
cancelClass : OperationSelection
supplementaryClass : OperationSelection
Tap
Tap
Search for class supplementary information
Search for class cancellation information

Figure 3. Activity Diagram for a Use Case

classified into *InputWidget* and *OutputWidget* and both of become components in a class diagram corresponding to a use case.

The Gesture class has a role of expressing requests for the operability of a system.

The State class is used to express a distinction of a property of a process in an early stage of development. Background processing has API usage restrictions.

The ExternalSystem class expresses a system with which an application can cooperate to improve the service. The class becomes an object node in an activity diagram and an actor of the use case. These classes include not only cooperating with other applications, but also the use of various types of hardware and communication methods.

C. *UML Models*

A use case diagram includes several use cases with the related actor, such as a user or available external application or hardware component. A developer may decide a root use case by relating the other use cases using *extend* or *include* relationships. The root use case represents a scenario of starting the application. By the end of the operation, each use case is defined by an activity diagram and the relationship is expressed by calling a sub activity corresponding to the other use case in the activity diagram (Figure 3).

An activity diagram expresses a series of processing actions with related data. The background action is distinguished from the foreground action by the use of a partition. An object node is used to denote the linking of external applications or hardware. A *User* partition includes user actions with input data whereas the *Interaction* partition includes actions with output data through a user interface.

Figure 4 shows a class structure of system partition specified by *State* of the glossary. In this model, the general components in an activity diagram are specified by smartphone features. A class of the glossary is displayed in red and a developer can design smartphone features by using this class.

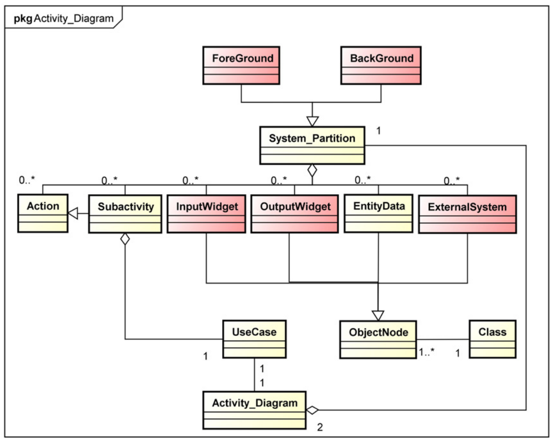

Figure 4. System Partition in Activity Diagram

Figure 5 shows how a use case corresponds to a screen of the application preventing complication of models. Information about input/output data that are used in the use case is expressed by a class diagram composed of a class corresponding to three types of classes in the glossary. Entity data is also defined by a class related to the use case, as shown in Figure 5.

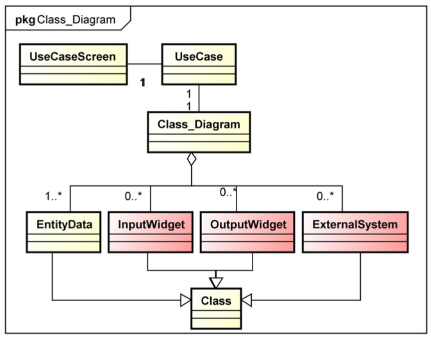

Figure 5. Use case and Class Diagram

D. *UI Design with GUI Builder*

Based on a use case diagram, a developer can design the UI screen image using the drag-and-drop WYSIWYG editor. In this step, the UI is designed using subclasses of *View* and *Gesture* in the glossary. As Figure 6 shows, there are 17 types of widgets in *View*. Each *View* has one or more *Gestures* that is a trigger to call the use case function. *EventAction* objects can have a connection with the other use cases. *EntityData* object expresses data that is created by the function and will be read by the *View* object. *Abstract GUI Information* is automatically generated or updated by these operations on the GUI builder. This sequence of operations corresponds to a sequence of actions in the related activity diagram.

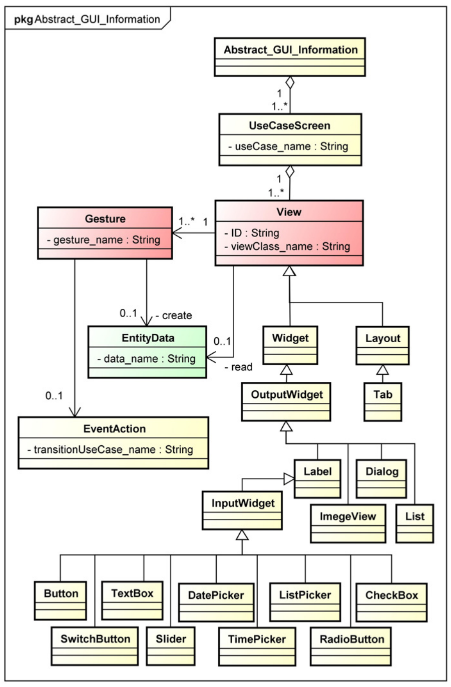

Figure 6. Abstract GUI Information

A developer defines attributes or values such as the size of a widget, the position, font type and font size, the content of the message presented, and a name of a label or button. Such data is saved as *Concrete GUI Information* written in XML.

Another important role of the GUI builder is to decide the most appropriate screen transition based on the amount of information caused by the combination of use cases. Figure 7 shows an example of how such a decision is made, to integrate two screens into one screen or not to integrate the screens.

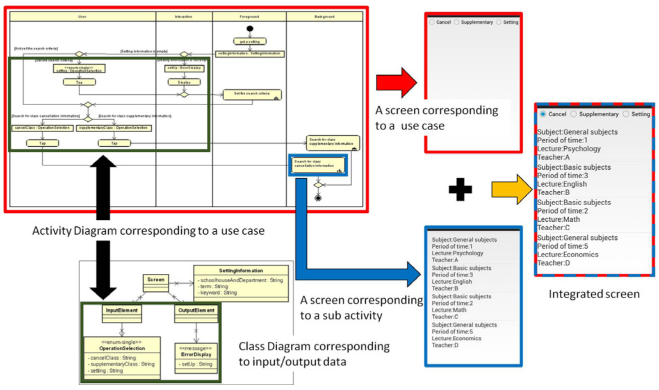
A screen corresponding to a use case
Activity Diagram corresponding to a use case
A screen corresponding to a sub activity
Class Diagram corresponding to input/output data
Integrated screen
Cancel
Supplementary
Setting
Screen
InputElement
OutputElement
SettingInformation

Figure 7. Products of UML-based MDD

IV. RELATED WORK

Lettner et al. [10] has stated that MDD is a promising approach for mobile phones in solving problems of conventional code-centric development approaches. They discuss the problem from the viewpoint of the reusability of parts of a system and the adaptability to various changing platforms. However, Lettner did not propose a concrete mechanism in which we can design reusable models with smartphone specific features independent of specific OSs.

There have been several studies of MDD for smartphone applications. In one study, Sabraou, et al. [11] proposed a MDD method to design GUI using object diagrams in UML. These diagrams are translated into XML based data on the Android GUI Meta model. However, in comparison with our approach, a developer cannot design the user interface in an intuitive manner. Moreover, consideration of screen transition in accordance of the amount of information is not discussed in the Sabraou study.

In another study, Diep et al. [12] proposed an MDD environment to provide developers with a platform-independent GUI design for mobile applications. Though the static screen composition can be defined, dynamic screen changes cannot be performed. In contrast, we use a GUI builder to design GUI. Moreover, we analyze application logic called by UI components using the activity diagram and the entity data. This combination ensures that dynamic screen changes can be performed.

MD2 [13] is a framework for cross-platform model-driven mobile development. In their approach, a developer needs to design an application model by using a specific DSL in text form. However, the DSL is insufficient to flexibly design the smartphone application model from both the structural view and the UI view.

Franzago et al. [14] also proposed a collaborative framework for the development of data-intensive mobile applications exploiting MDD techniques and separation of concerns. Our approach uses familiar modeling Language UML and GUI builder which can easily use in intuitive manner.

V. Conclusion and Future Work

In our paper, we proposed an MDD method by using an existing UML modeling tool and our own GUI builder to operate abstract widgets in an intuitive manner. This allows flexibility in the design of the smartphone application from both the structural view and the UI view. We are currently developing the GUI builder using the Android tablet PC based on the Smartphone Feature Specific Model. By applying our method to more smartphone applications, we will verify if minute differences between features of OSs can be discussed on the model.

References

[1] OMG," Unified Modeling Language", http://www.uml.org/ (accessed: Aug. 13, 2014)

[2] S. J. Mellor, K. Scott, A. Uhl, and D. Weise, MDA Distilled Principles of Model-Driven Architecture. Addison-Wesley, 2004.

[3] OMG, "MDA Guide Version 1.0.1." Object Management Group, Tech. Rep., 2003.

[4] ADT Plugin, http://developer.android.com/tools/sdk/eclipse-adt.html (accessed: Aug. 13, 2014)

[5] astah:http://astah.change-vision.com/ja/ (accessed: Aug. 13, 2014)

[6] I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard. Object-oriented software engineering: A usecase driven approach, Addison-Wesley Publishing, 1992.

[7] Android developers, http://developer.android.com/index.html, 2014

[8] iOS Developer Library, https://developer.apple.com/library/ios/navigation/ (accessed: Aug. 13, 2014)

[9] Windows Phone Dev Center, https://dev.windowsphone.com/en-us/home (accessed: Aug. 13, 2014)

[10] M. Lettner and M. Tschernuth "Applied MDA for Embedded Devices: Software design and code generation for a low-cost mobile phone", the 34th Annual IEEE Computer Software and Applications Conference Workshops, 2010, pp. 63-68.

[11] A. Sabraou, M. E. Koutb, and I. Khriss, "GUI Code Generation for Android Applications Using a MDA Approach", International Conference on Complex Systems, 2012, pp. 1-6.

[12] C. K. Diep, Q. N. Tran and M. T. Tran, "Online Model-driven IDE to Design GUIs For Cross-platform Mobile Applications", SolCT, ACM International Conference Proceeding Series, 2013, pp. 294-300.

[13] H. Heitkotter, A. T. Majchrzak and K. Herbert. "Cross-platform model-driven development of mobile applications with md 2." Proceedings of the 28th Annual ACM Symposium on Applied Computing. ACM, pp. 405–411, 2013.

[14] M. Franzago, H. Muccini and I. Malavolta, Towards a collaborative framework for the design and development of data-intensive mobile applications. In Proceedings of the 1st International Conference on Mobile Software Engineering and Systems (MOBILESoft 2014). pp. 58-61, 2014.