Tai-hoon Kim Hojjat Adeli
Haeng-kon Kim Heau-jo Kang
Kyung Jung Kim Akingbehin Kiumi
Byeong-Ho Kang (Eds.)

Communications in Computer and Information Science 257

Software Engineering, Business Continuity, and Education

International Conferences ASEA, DRBC and EL 2011,
Held as Part of the Future Generation
Information Technology Conference, FGIT 2011,
in Conjunction with GDC 2011,
Jeju Island, Korea, December 2011, Proceedings

Springer

A Study on UML Model Convergence Using Model Transformation Technique for Heterogeneous Smartphone Application

Woo Yeol Kim, Hyun Seung Son, and Robert Young Chul Kim

Dept. of CIC(Computer and Information Communication), Hongik University,
Jochiwon, 339-701, Korea
{john,son,bob}@selab.hongik.ac.kr

Abstract. Smart phones have various types of platform such as Android, Cocoa touch, and Windows Phone. As software is developed in one specific platform, it is impossible to use this software on different platforms. To solve this problem, this paper suggests UML model convergence with model conversion method to develop heterogeneous software per each platform. The suggested method consists of two stages: one TIM(target independent model) stage to abstract a model independent on the particular platform and other TSM(target dependent model) stage to convert the independent model into several target models based on the Model-to-Model transformation method. As a case study, a calculator model on Android oriented Platform is converted into another model on Windows oriented Platform.

Keywords: Model Transformation, Model Convergence, UML, Heterogeneous Smartphone Application, Cross Platform.

1 Introduction

Many different development platforms included in smart phones, such as Symbian, OpenC, iPhone, Android, Windows Phone, and Palm operating systems contains various technologies such as widget, Web runtimes, Python, Lazarus, Brew, Java Mobile Edition (ME), .NET Compact Framework (CF), and Flash Lite [1]. These provide strong mobile contents such as audio, video, multimedia messaging, and Flash. For this reason, most software developers prefer the particular platform-based development. However, as software is developed based on a specific platform, it is impossible to reuse the software into other platforms.

In this paper, we adapt MDD (Model-Driven Development) approach [2] to develop heterogeneous software. The original MDD needs to make automation of the process from design model to code generation [3]. In this method, it is possible to convert one upper model to different lower models based on a basic top-down mechanism. With a platform independent model (upper model), it may generate several platform dependent models. Then model convergence through free movement with common elements between and among heterogeneous models might be difficult. In the previous studies [4,5,6,7,8], smart platform development was conducted with applied MDD mechanism, but it was not for model convergence. This paper suggests a way for model convergence for improving such results

The suggested method consists of two stages: one TIM(target independent model) stage to abstract a model independent on the particular platform and other TSM(target dependent model) stage to convert the independent model into several target models based on the Model-to-Model transformation method. This paper is organized as follows. Chapter 2 describes Model Transformation with related studies. Chapter 3 mentions model convergence for heterogeneous platforms. Chapter 4 presents a case study. Finally, Chapter 5 makes a conclusion.

2 Related Studies

The existing methods to convert models can be largely classified with Direct-Manipulation, Relational, Graph-Transformation, Structure-Driven, and Hybrid approaches [9]. The Direct-Manipulation approach provides internal model conversion and control API. This approach is also good in that there is no constraint on conversion. But its weakness is in that all parts must be materialized for conversion. The relational approach defines a constraint on the relationship between elements of the source model and the target model. Also while there are various connection methods on mapping rules, it is difficult to prepare for a conversion language. The Graph-Transformation approach uses graphs for easily understanding it. However, most conversions are complex, which is a weak point of these approaches. The Structure-Driven method is to provide meta-model definitions of each source and target model with model element structures. This weak point applies to the same model conversion. The Hybrid approach is a combination with two or more these approaches based on different strengths and weaknesses. The Hybrid approach enables various types of conversion, which may make the conversion process complex.

3 Model Conversion Methods for Heterogeneous Platforms

The basic model conversion for heterogeneous platforms is as shown in Figure 1. A model converted is selected in the model on an 'A' platform while the Model-to-Model transformation is applied to convert it into a 'B' Platform model.

The important technology for model convergence is the model-to-model conversion. The description for this is displayed in Figure 2. First, the model to be converted is selected. The selected model includes a library subject to the platform. What is important at this point is the process to separate a model dependent on the platform from that independent of the platform. The method to separate for design is described in detail with four example questions. In this paper, the process to convert this platform-dependent model into a platform-independent model will be called abstractization. This is applied because the conversion into platform-independent

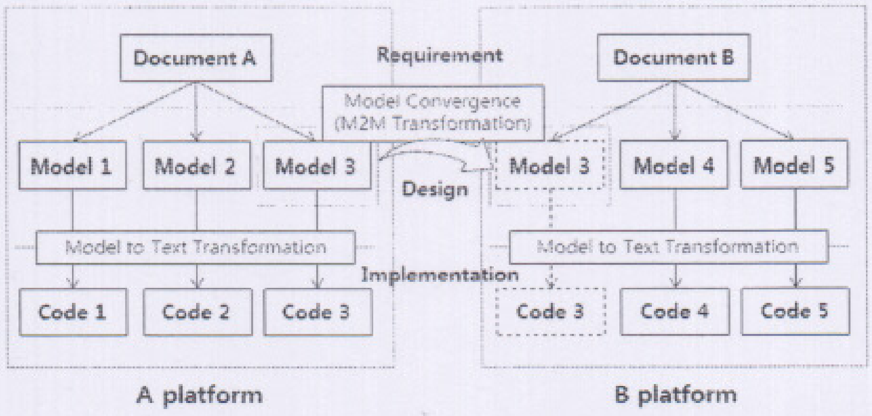

Fig. 1. Schematic diagram for model convergence

models is easier that that into the existing platform-dependent models. Also the abstracted model is converted into a platform subject to conversion. The conversion rule is prepared based on transformation language. The model generated finally through the conversion process is inserted into the model subject to convergence.

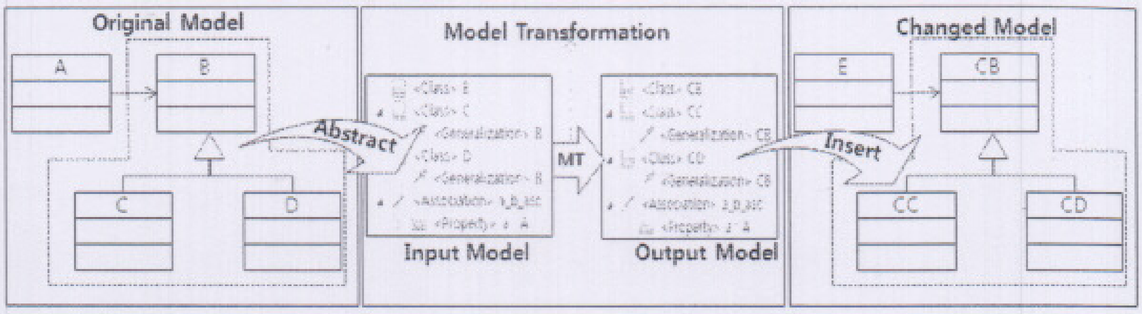

Fig. 2. Model-to-Model Transformation for Model Convergence

4 Case Study

The application model for Android platform, just like Figure 3, is prepared as a class diagram. The calculator class is the main class to play a role of calculators. The event processing in Android platform applies *Listener*. *ButtonListener* class was used to process many buttons at one time. *ResultViewer* class is a class to show the calculated result. The behavior processing when the calculator button is pressed is *ButtonAction* class. As for *ButtonAction* class, the roles were divided into *Backspace*, *Clear*, *ClearEach*, *Dot*, *Number*, and *Operation* classes. *Backspace* is a class to delete one letter when the "BS" button is pressed. *Clear* is a class to delete all data when the "C" button is pressed. *ClearEach* is a class to delete a formula when the "CE" is pressed. *Dot* is a class to feed decimal points when the "." button is pressed. *Number* is a class carrying out processing when number keys are pressed and *Operation* is a class applied when " + ", " - ", " * ", " / " keys are pressed.

Figure 4 is a diagram displaying the results from model conversion. Based on the model conversion process, the result shows that most of the independent model

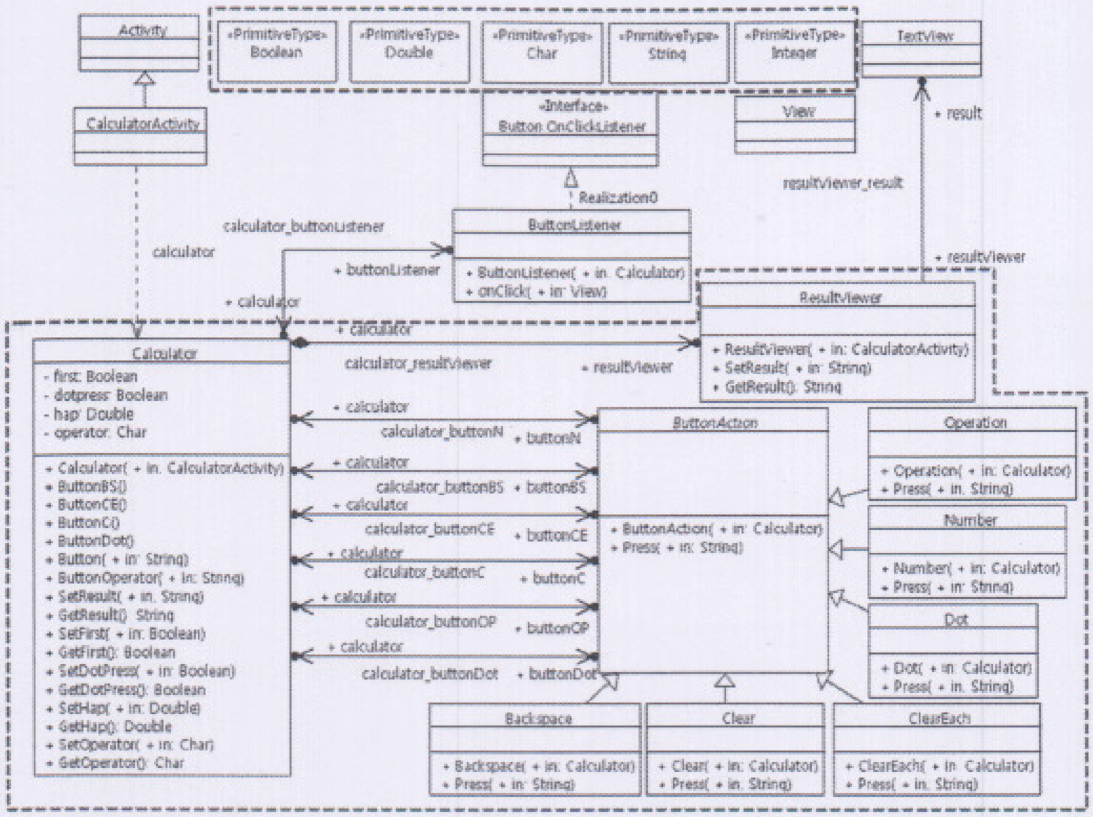

Fig. 3. Class Diagram of Android Platform

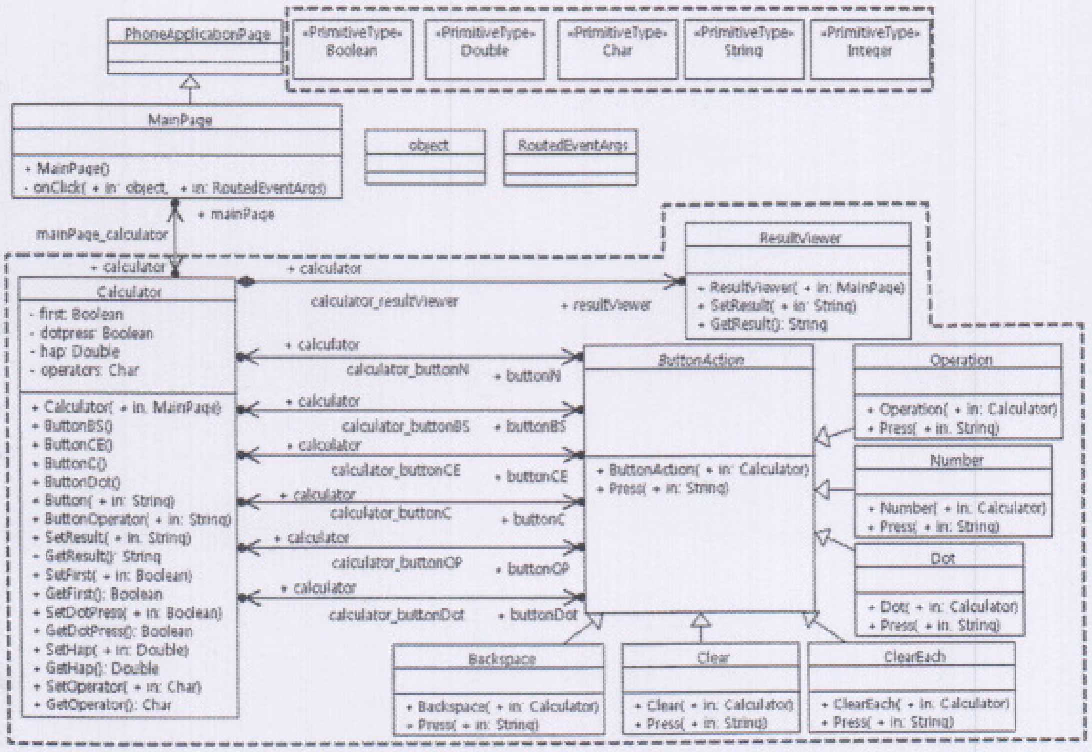

Fig. 4. Class Diagram of Windows Phone Platform

structures can be reused. Also the generation of *MainPage* and *PhoneApplicationPage* classes that are dependent on Windows Phone based on model conversion can be confirmed in the figure. As for Windows Phone, *MainPage* directly receives Event through Method and there is no *ButtonListener* class. In order to increase the reuse of models upon convergence, we can tell that based on the case study, it is important to separate platform-independent areas from platform-dependent ones upon designing model.

5 Conclusion

In the smart phone environment, the model convergence for heterogeneous platforms is more necessary than that for homogeneous ones. In this paper, for the model convergence in heterogeneous platform environment, the original MDD (Model-Driven Development) was adopted into smart phone area. The MDD is a method that automates the process from a software design model to software materialization, and enables the conversion of one upper model into lower models of different types. MDD is a very appropriate way for model conversion, but is not capable of horizontal movement between heterogeneous models. Thus, it is not advantageous in terms of model convergence. Therefore, this paper has suggested a method for model convergence that applies the Model Transformation.

The suggested method uses the Model Transformation as a major technology of MDD and conducts convergence of the existing model with the target model. The first stage is abstractization, separating the platform-dependent models from those platform-independent ones. The second stage is Model-to-Model Transformation, which converts the model from the abstracted model based on the correlation generation rules and class-generating templates into a subject model. The model generated based on abstractization is platform-independent while the dependent attribute of the subject model to be converted can be generated without revising the existing model. As such generation is repeated, we can create class-generating templates and correlation generation methods.

Acknowledgments. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2011-0004203) and the Ministry of Education, Science Technology (MEST) and National Research Foundation of Korea(NRF) through the Human Resource Training Project for Regional Innovation.

References

1. Gavalas, D., Economou, D.: Development Platforms for Mobile Applications: Status and Trends. IEEE Software 28(1), 77–86 (2011)
2. Selic, B.: The pragmatics of model-driven development. IEEE Software 20(5), 19–25 (2003)

3. Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches. IBM Systems Journal 45(3), 621–645 (2006)
4. Kim, W.Y., Son, H.S., Kim, J.S., Kim, R.Y.C.: Development of Windows Mobile Applications using Model Transformation techniques. Journal of KIISE: Computing Practices and Letters 16(11), 1091–1095 (2010)
5. Kim, W.Y., Son, H.S., Kim, R.Y.C.: Design of Code Template for Automatic Code Generation of Heterogeneous Smartphone Application. In: Kim, T.-h., Adeli, H., Robles, R.J., Balitanas, M. (eds.) ACN 2011. CCIS, vol. 199, pp. 292–297. Springer, Heidelberg (2011)
6. Kim, W.Y., Son, H.S., Kim, J.S., Kim, R.Y.C.: Adapting Model Transformation Approach for Android Smartphone Application. In: Kim, T.-h., Adeli, H., Robles, R.J., Balitanas, M. (eds.) ACN 2011. CCIS, vol. 199, pp. 421–429. Springer, Heidelberg (2011)
7. Kim, W.Y., Son, H.S., Yoo, J., Park, Y., Kim, R.Y.C.: A Study on Target Model Generation for Smartphone Applications using Model Transformation Technique. In: International Conference on Internet (ICONI) 2010, vol. 2, pp. 557–558 (2010)
8. Son, H.S., Kim, W.Y., Woo Sung J., Kim, R.Y.C.: Development Android Application using Model Transformation. In: Joint Workshop on Software Engineering Technology 2010, vol. 8(1), pp. 64–67 (2010)
9. Czarnecki, K., Helsen, S.: Classification of model transformation approaches. In: OOPSLA 2003 Workshop on Generative Techniques in the Context of Model-Driven Architecture (2003)