Tai-hoon Kim
Hojjat Adeli
Rosslin John Robles
Maricel Balitanas (Eds.)

Communications in Computer and Information Science 199

Advanced Communication and Networking

Third International Conference, ACN 2011
Brno, Czech Republic, August 2011
Proceedings

Springer

Adapting Model Transformation Approach for Android Smartphone Application*

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

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

Abstract. The current smart phone applications are developed subordinately to its platform. In other words, applications are developed according to its own unique platform by each relevant vender, such as Cocoa for Apple, Android for Google and Windows Mobile for Microsoft. This paper suggests adapting the whole process of MDD (Model Driven Development) for the development of heterogeneous smart applications. Our proposed MDD is composed of Model-to-Model Transformation, which separates the independent from the dependent platform and transforms the independent model to the dependent model through rules on model transformation, and Model-to-Text Transformation, which forms codes from each dependent model. Model, metamodel and model transformation language are required to achieve Model-to-Model Transformation while model, metamodel and code template are needed for Model-to-Text Transformation. UML model, UML metamodel, ATL model transformation language and Acceleo code generation language also are adapted MDD in smart phone development environment. This paper also mentions a case of applying MDD in SnakePlus game.

Keywords: Model Transformation, Android, Smartphone, ATL(ATLAS Transformation Language), Metamodel.

1 Introduction

Reusable system for achieving maximum use of previously formed resources must be established to achieve quick software development. However, reuse of software is difficult to achieve as the mobile-embedded software is dependent on the system and is developed centered on the source code [1]. Furthermore, as various platforms are provided by different cell phone manufacturers and suppliers, a method for solving this problem is required.

* This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency)(NIPA-2011-(C1090-1131-0008)) and the Ministry of Education, Science Technology (MEST) and the IT R&D Program of MKE/KEIT [10035708, "The Development of CPS(Cyber-Physical Systems) Core Technologies for High Confidential Autonomic Control Software"].

T.-h. Kim et al. (Eds.): ACN 2011, CCIS 199, pp. 421–429, 2011.

MDD (Model Driven Development) is the mechanism for changing necessary technical model after designing platform-independent metamodel to automatically form code through the model. The model can be selected to re-create code in case the application program is required in another environment. Revision of application program is not required. This method can increase the productivity of code related with model re-use [2-4]. Thus, one model can be automatically formed into heterogeneous models by applying the model transformation technique in smart phone development environment.

This paper applies MDD in Android platform to develop heterogeneous smart phone applications. To apply in Android platform, application structure is analyzed and classified into independent and dependent characteristics. MDD is classified into Model-to-Model Transformation and Model-to-Text Transformation [5]. Model-to-Model Transformation uses ATL(ATLAS Transformation Language)[6] while Acceleo [7] is used in Model-to-Text Transformation. The entire process, from classification of independent/dependent model to code generation, is automatically executed by using tools. As an example, MDD is applied in the development process of the game, SnakePlus. The application of model transformation technique in smart phone environment can be verified through the application case. Furthermore, the application case presented the possibility of applying the technique in other various platforms, such as iPhone and Window Mobile.

This paper is organized as follows. Chapter 2 explains related studies, including the basic concept of MDD, model transformation method and APT. Chapter 3 mentions the MDD-based development method of Android application. Chapter 4 describes Android application in the development process. Lastly, Chapter 5 mentions the conclusion and future work.

2 Related Work

MDD is classified into Model-to-Model Transformation and Model-to-Text Transformation. Model-to-Model Transformation is the method of transforming input source model as the subject model. Model transformation can be achieved from model to model, model to code, and code to model. Furthermore, transformation can be achieved in various models, such as UML, control flow diagram, and data flow diagram. Although the formation of metamodel is not required as UML metamodel is provided in UML, MOF(Meta Object Facility)[8] is used to design metamodel in models without metamodel.

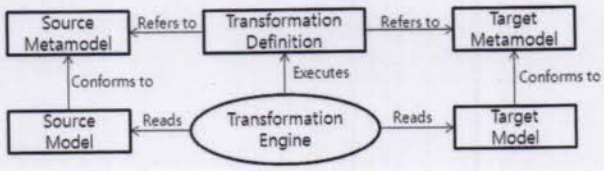

Fig. 1. Basic principle of model transformation [9]

The basic principle of model transformation is presented in Figure 1. To execute model transformation, source model, source metamodel, target metamodel, and transformation must be defined. Model transformation engine uses such factors to transform into target model. Language that defines transformation rules is an important factor in model transformation. UMT(UML Model Transformation)[10], MTL(Model Transformation Language)[11], QVT(Query / View / Transformation)[12], ATL(ATLAS Transformation Language)[13,14] are the types of languages. UMT is the method using XML, XMI[15], XSLT. XMI is the input value in model transformation. However, as this is fundamentally XML, transformation is achieved by XSLT, the transformation language of XML. MTL can achieve model transformation of source models written in DSL(Domain Specific Language) and use the concept of Java virtual machine to transform metamodels written in metamodel language as a type of compiler. As the model transformation language selected as the standard in OMG, QVT is advantageous in that it can accurately express the specific transformation point, such as by When, Where, and can also form and re-use patterns. Like MTL, ATL executes model transformation using MDA framework by transformation definition language and transformation engine. The definition of transformation rule of MTL is more accurate and easier to understand in ATL, which can also define complex transformation rules. Furthermore, it is advantageous in that it provides re-use and synthesis of transformation models. Among the diverse types of transformation languages, this thesis selected ATL for such advantages.

Model-to-Text Transformation is the method of forming code from model. This method is classified into two types: visitor-based and template-based [9]. In the visitor-based code generation method, the visitor mechanism is composed of purifying the internal expression of model and writing text in text stream. An example of this method is Jamda, the object-oriented framework that provides the assembly of classes that represent UML model [16].

Template-based method is more similar with the code than the visitor mechanism. This is easily used in the repeated development of template. Template can represent text work due to the approach presented in section as well as code pieces within text that are incorrect in syntax or meaning. Acceleo is an example of this method. Template-based Acceleo is used for code generation in this paper.

3 Model Transformation for Smartphone Application

MDD can be classified into Model-to-Model and Model-to-Text. Although the final goal of model transformation method is to form code from model, the middle model can be used to achieve greater modulation and maintenance, optimization and tuning, or to eliminate defects. Furthermore, Model-to-Model transformation is helpful in calculating and uniting different views of system module. Model-to-Text is focused on transforming the relevant model into code.

Figure 2 presents the outline diagram of MDD-based development method, the method used in this paper. Source & Target Metamodel, TIM(Target Independent Model), and transformation spec must be entered to execute Model-to-Model, and ATL tool is required for execution. Through this input process, TIM can be automatically formed through TSM(Target Specific Model). TIM is the independent

model of the development target while TSM refers to the dependent model of the target. Model, metamodel, and model transformation language is required for Model-to-Model Transformation. UML model, UML metamodel, and ATL model transformation language were selected for use.

TSM and transformation spec must be entered to execute Model-to-Model, and Acceleo tool is required for execution. TSM Code is formed through this input process. Model, metamodel, and code template are required for Model-to-Text Transformation. Like Model-to-Model Transformation, UML was used in model and metamodel while Acceleo was used for code template to apply MDD in smart phone development environment.

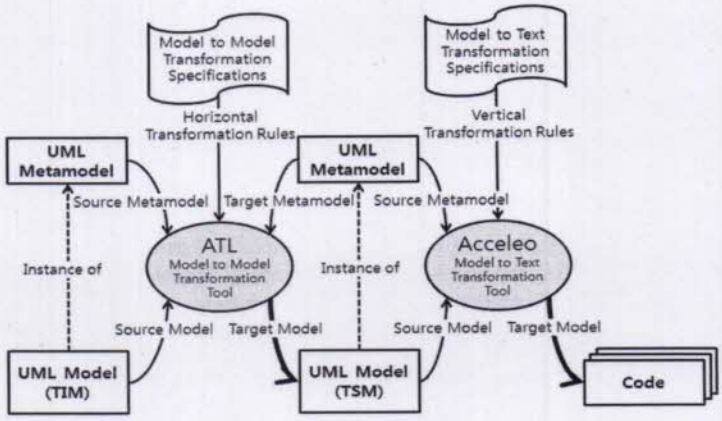

Fig. 2. Outline of model transformation method

4 Case Study

SnakePlus is the game application created based on the existing Snake[17] game. The existing Snake game is progressed by moving the snake North, East, West, and South to eat apples. In SnakePlus, the snake can be moved in a total of 8 directions and various items have been added to increase the number of entertaining elements in the previously simple game. This paper explains the model transformation process based on SnakePlus.

4.1 TIM(Target Independent Model)

In this example, one target independent model (TIM) is used to form TSM and code for Android. SnakePlus is composed of a total of 7 packages: Main, StageFactory, StageComponent, Background, Item, Wall, and Snake. Main initializes SnakePlus and draws image within the screen. Stage is formed and image information of objects is loaded on memory at the beginning of SnakePlus. StageFactory manages the stage and objects of SnakePlus. Objects are loaded on memory and location information is managed. StageComponent manages the common information of objects within

SnakePlus. The function of reading and writing the coordinates of object and image information of object are connected to ComponentManager. BackGround manages the background information within stage. It also achieves input and output of images in background information. Item manages the coordinates and image information of SnakePlus items. Item includes obstacle, apple, poison, and mouse. It also achieves input and output of item coordinates and image information. Wall manages the wall information within stage. Furthermore, it achieves input and output of wall images and coordinates. Snake manages information of snake within stage and achieves input and output of snake location and direction as well as image information.

Main package is composed of StartClass, SnakePlusView, SnakPlusMain, ComponentManager, RefreshHandler class and all information is initialized when starting SnakePlus. Main package is as presented in Figure 3. The following presents explanations on Main package.

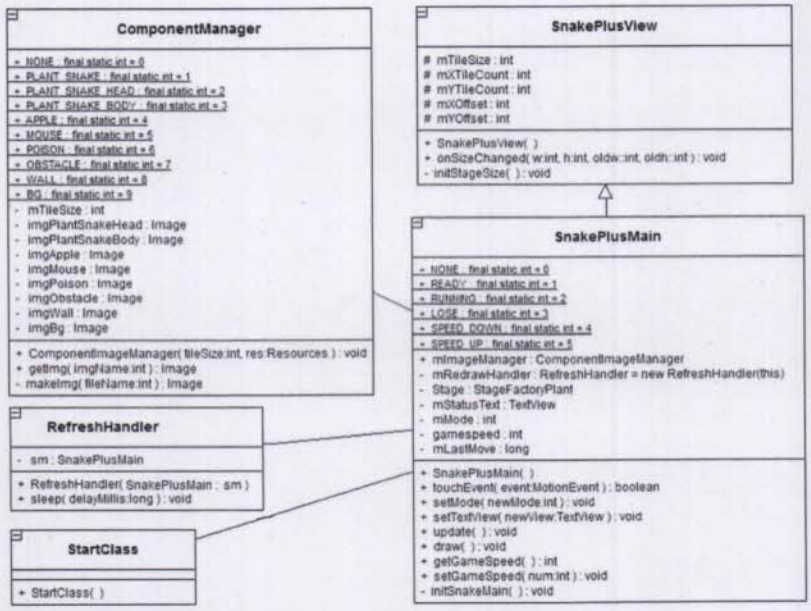

Fig. 3. Class diagram of TIM

4.2 Model-to-Model Transformation (TSM)

Android SnakePlus is composed of a total of 7 packages: Main, StageFactory, StageComponent, BackGround, Item, Wall, Snake. Like TIM, Main package is the starting point of Android TSM. However, the class composition within package alters due to characteristics of Android platform. Android can be written in XML and Java languages and its structure can be largely classified into layout and activity. Layout is XML file and button, textbox directly displayed on screen are written by XML code. Thus, it is the file for writing buttons that are directly released on smart phone screen.

Activity is class file and codes, such as event listener of layout, are written. Activity is the starting point of Android. After the activity is started, it immediately summons the layout. Because of this, StartClass of TIM transforms to SnakePlusActivity class holding activity properties and also changes the class-composing method. The composition of classes other than SnakePlusActivity class is identical to TIM.

Main package is composed of SnakePlusActivity, SnakePlusView, SnakePlusMain, ComponentManager, RefreshHandler class. When compared with the structure of TIM, it presents differences in the part where StartClass class is changed to SnakePlusActivity class, the part displaying change in the image saving form of ComponentManager class, and the part where the method of SnakePlusMain class is changed to Android interior API. Figure 4 presents a detailed diagram on such differences.

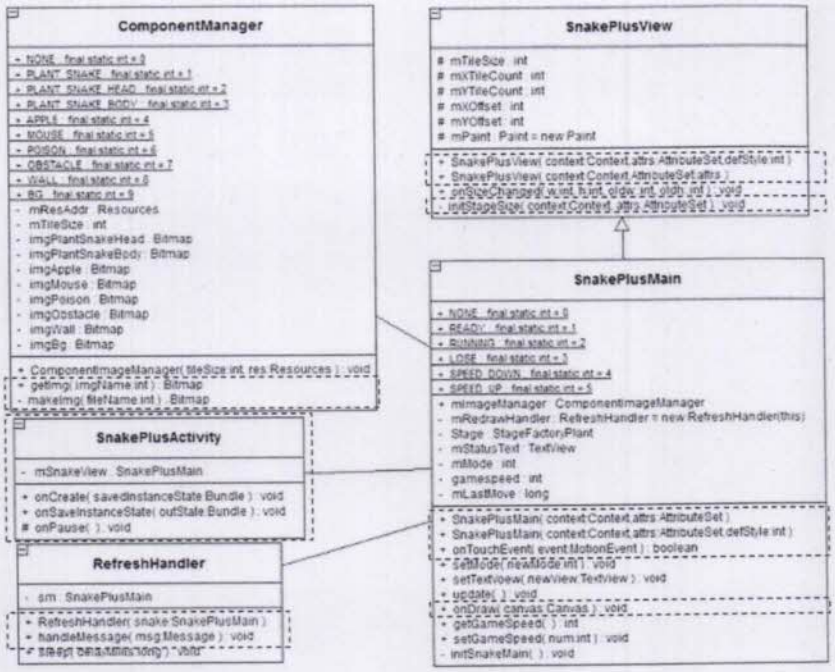

Fig. 4. Class diagram of TSM for Android

4.3 Model-to-Text Transformation (TDC)

Code generation template is required to achieve formation of Android code. This code template uses the Java code generation template, which was defined in the previous example. As observed in the target-dependent model, a total of 19 classes exist in the formed code. Among these classes, StartClass of target-independent model is SnakePlusActivity, while draw method of SnakePlusMain class is changed to onDraw and touchEvent is changed to onTouchEvent. Figure 5 is the formed SnakePlusMain.

```
public class SnakePlusMain extends SnakePlusView {
  private RefreshHandler mRedrawHandler = new RefreshHandler(this);
  private StageFactoryPlant stage;
  private TextView mStatusText;
  private int gameSpeed;
  private long mLastMove;
  private int mMode;
  public final static int NONE = 0;
  public final static int READY = 1;
  public final static int RUNNING = 2;
  public final static int LOSE = 3;
  public final static int SPEED_DOWN = 4;
  public final static int SPEED_UP = 5;
  ComponentImageManager mImageManager;
  public SnakePlusMain(Context context, AttributeSet attrs) {
  }
  public SnakePlusMain(Context context, AttributeSet attrs,
int defStyle){
  }
  private void initSnakeMain() {
  }
  public boolean onTouchEvent(MotionEvent event) {
  }
  public void setTextView(TextView newView) {
  }
  public int getGameSpeed(){
  }
  public void setGameSpeed(int num){
  }
  public void update(){
     invalidate();
  }
  public void onDraw(Canvas canvas) {
     super.onDraw(canvas);
  }
}
```

Fig. 5. Class code of formed SnakePlusMain

4.4 Results of Execution

To complete program, additional codes for function must be written after forming code of target-dependent model. Results presented in Figure 6 can be checked when this program is compiled and executed.

The development of the final application could be achieved through target-independent model, target-dependent model, and code generation automation process. Although perfect code generation cannot be achieved currently, approximately 45% of Windows mobile application codes can be formed through UML. Fast execution of design and realization process can be achieved through increased code generation rate.

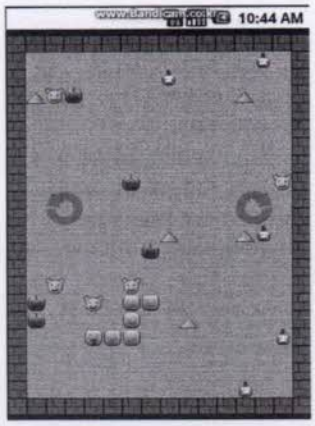

Fig. 6. Results of execution

5 Conclusion

This paper applied MDD in Android platform development to achieve heterogeneous smart phone development. For MDD application, classification was made between Model-to-Model Transformation and Model-to-Text Transformation and automation tool was used to achieve development. The application structure of Android platform was analyzed to execute Model-to-Model Transformation. By using the results of analysis, independent and dependent characteristics were classified and rules of model transformation were created based on independent/dependent models. ATL was used to describe the rules of model transformation and model transformation was executed in eclipse environment. In result, it was verified that class diagram was formed according to the rules of model transformation.

Java used in Android was analyzed to execute Model-to-Text Transformation. Code template was written based on the results of analysis. It was written in Acceleo grammar to execute transformation in eclipse environment. In result, it was verified that code was formed according to rules of model transformation.

The possibility of achieving transformation from formed target-independent model to other heterogeneous platform was presented by application in SnakePlus of Android environment. Transformation to heterogeneous platforms, such as iPhone, Window mobile, can be achieved by using the platform independent model of this paper and re-defining the rules of model transformation. In future research, iPhone and Window mobile platform will be analyzed to apply the proposed method in each platform.

References

1. Jantsch, A.: Modeling Embedded System and SOCs. Morgan Kaufmann, San Francisco (2004)
2. Kim, W.Y., Son, H.S., Park, Y.B., Park, B.H., Carlson, C.R., Kim, R.Y.C.: The Automatic MDA (Model Driven Architecture)Transformations for Heterogeneous Embedded Systems. In: Proceedings of The 2008 International Conference on Software Engineering Research and Practice, vol. 2, pp. 409–414 (July 14, 2008)
3. Kim, W.Y., Kim, R.Y.C.: A Study on Modeling Heterogeneous Embedded S/W Components based on Model Driven Architecture with Extended xUML. The KIPS Transactions 14-D(1) (February 2007)
4. Kim, W.Y., Son, H.S., Kim, R.Y.C., Carlson, C.R.: MDD based CASE Tool for Modeling Heterogeneous Multi-Jointed Robots. In: Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering, vol. 7, pp. 775–779 (April 01, 2009)
5. Czarnecki, K., Helsen, S.: Classification of model transformation approaches. In: OOPSLA 2003 Workshop on Generative Techniques in the Context of Model-Driven Architecture (2003)
6. Wikipedia, ATLAS Transformation Language, `http://en.wikipedia.org/wiki/ATLAS_Transformation_Language`
7. Obeo, Acceleo User Guide, `http://www.acceleo.org/`
8. OMG, Meta Object Facility Specification. In: OMG Unified Modeling Language Specification, Version 2.0 (January 2006)
9. Czarnecki, K., Helsen, S.: Feature-Based Survey of Model Transformation Approaches. IBM Systems Journal 45(3), 621–664 (2006)
10. Grønmo, R., Oldevik, J.: An Empirical Study of the UML Model Transformation Tool(UMT). In: The First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA), Geneva, Switzerland (February 2005)
11. Vojtisek, D., Jézéquel, J.-M.: MTL and Umlaut NG:Engine and Framework for Model Transformation, `http://www.ercim.org/publication/Ercim_News/enw58/vojtisek.html`
12. OMG, Documents associated with Meta Object Facility (MOF) 2.0 Query / View / Transformation, Version 1.0 (April 2008)
13. Bézivin, J., Dupé, G., Jouault, F., Pitette, G., Rougui, J.E.: First Experiments with the ATL Model Transformation Language: Transforming XSLT into XQuery. In: Proceedings of the Workshop on Generative Techniques in the Context of Model Driven Architecture, Anaheim,CA (2003)
14. Jouault, F., Kurtev, I.: Transforming Models with ATL. In: Proceedings of Model Transformations in Practice Workshop (MTIP), MoDELS Conference, Montego Bay, Jamaica (2005)
15. OMG, XML Metadata Interchange Specification. In: OMG Unified Modeling Language Specification, Version 2.1.1 (December 2007)
16. Jamda, The Java Model Driven Architecture 0.2 (2003), `http://sourceforge.net/projects/jamda/`
17. Snake, Technical Resources, `http://developer.android.com/resources/samples/Snake/`

Communications in Computer and Information Science

The CCIS series is devoted to the publication of proceedings of computer science conferences. Its aim is to efficiently disseminate original research results in informatics in printed and electronic form. While the focus is on publication of peer-reviewed full papers presenting mature work, inclusion of reviewed short papers and abstracts reporting on work in progress is welcome, too. Besides globally relevant meetings with internationally representative program committees guaranteeing a strict peer-reviewing and paper selection process, conferences run by societies or of high regional or national relevance are considered for publication as well.

The topical scope of CCIS spans the entire spectrum of informatics ranging from foundational topics in the theory of computing to information and communications science and technology and a broad variety of interdisciplinary application fields.

Publication in CCIS is free of charge. No royalties are paid, however, CCIS volume editors receive 25 complimentary copies of the proceedings. CCIS proceedings can be published in time for distribution at conferences or as post-proceedings, as printed books and/or electronically as CDs; furthermore CCIS proceedings are included in the CCIS electronic book series hosted in the SpringerLink digital library.

The language of publication is exclusively English. Authors publishing in CCIS have to sign the Springer CCIS copyright transfer form, however, they are free to use their material published in CCIS for substantially changed, more elaborate subsequent publications elsewhere. For the preparation of the camera-ready papers/files, authors have to strictly adhere to the Springer CCIS Authors' Instructions and are strongly encouraged to use the CCIS LaTeX style files or templates.

Detailed information on CCIS can be found at www.springer.com

ISSN 1865-0929

ISBN 978-3-642-23311-1

› springer.com