ISSN: 1738-9984

International Journal of Software Engineering and Its Applications

IJSEIA

Vol.7, No.3, May, 2013

SCIENCE & ENGINEERING
RESEARCH SUPPORT SOCIETY

International Journal of Software Engineering and Its Applications
Vol. 7, No. 3, May, 2013

SMTL Oriented Model Transformation Mechanism for Heterogeneous Smart Mobile Models

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

Dept. of CIC(Computer and Information Communication), Hongik University
Sejong Campus, 339-701, Korea
E-mail: {son, jskim, bob}@selab.hongik.ac.kr

Abstract

Until now, there are not existed any research to reuse any software on heterogeneous smartphones for interoperating between Android/iphone and iphone/Android. To do this, our previous approach [7, 8, 12, 13, 14, 15, 16] just used UML metamodel and model transformation language, ATL, based on model oriented Architecture/development (MDA/MDD) to embedded systems. But it has limited for Model Transformation language to represent all transformation rules with ATL. This limitation is not suitable for heterogeneous smartphone model transformation, and also impossible to extension of ATL. To solve this problem, this paper suggests Smartphone Model Transformation Language (SMTL) oriented model transformation mechanism for heterogeneous smartphone. We define SMTL which easily manipulates more input model in SMTL engine. Through invoking operation in SMTL engine, it is directly mapped with API in Eclipse modeling Framework (EMF). In addition, design to use XPath as XML technique instead of OCL to search data in source model.

Keywords: *Model Transformation Language, Smartphone, Heterogeneous, Metamodel, Transformation Engine*

1. Introduction

Recently, there are diverse software development platforms for smartphones such as Android [1, 2, 3], iPhone [4], and Windows Phone [5]. Nevertheless, it may be impossible to reuse the particular platform dependent software with the different programming language and application program interface (API) into other platform. Our first research focused on model driven development (MDD) to solve this problem of smartphone environment [7, 8]. Originally, MDD means to develop software with the conversion of platform independent model (PIM) to platform dependent model (PDM) for interoperating in Java, Corba, and .Net platform [9]. To use this mechanism absolutely needs to have model transformation technique that may automatically generate code with the converted model between PIM and PSM [10]. Our current research focuses on developing heterogeneous software how to adopt model transformation mechanism into smartphone software development, that is, a way simultaneously to develop software on different smart mobile environment such as Android, iPhone, and MS Window phone.

For example, we first applied window mobile application with ATLAS Transformation Language (ATL) [11] based on Model-Model Transformation, but limited to UML Class Diagram (CD) for the automatic conversion from platform independent model (PIM) to platform dependent model (PDM) [7], and confirmed to create the basic construct and change class and method. With this, it shows possibly to convert other dependent platforms with a platform independent one [12-16], which can be possibly transformed into Code with Acceleo based on Model-Text transformation. In this time, our research extends Message sequence

Diagram (MSD) with transformation mechanism [6]. On these researches, we found to require very complicated rules even though transforming a simple model with the existing ATL based model transformation. Therefore, we suggest SMTL (smartphone model transformation language) to develop heterogeneous smart mobile software. In this paper, we extend SMTL based on our first model language, multiple model transformation language (MMTL) suggested by Kim [12] for smartphone software. This SMTL is designed directly to link with API within Eclipse Modeling Framework (EMF) to manipulate metamodel more easily. The language is currently ongoing research and to extend model transformation tool.

Through invoke operation in SMTL engine, it is directly mapped with API in EMF(Eclipse modeling Framework). In addition, design to use XPath as XML technique instead of OCL to search data in source model.

The paper is organized as follows. Chapter 2 explains the basic concepts related to model transformations Language. Chapter 3 mentions the language constructs on SMTL (smartphone model transformation language). Chapter 4 describes case study support available for SMTL. Chapter 5 gives conclusion and future works.

2. Related Work

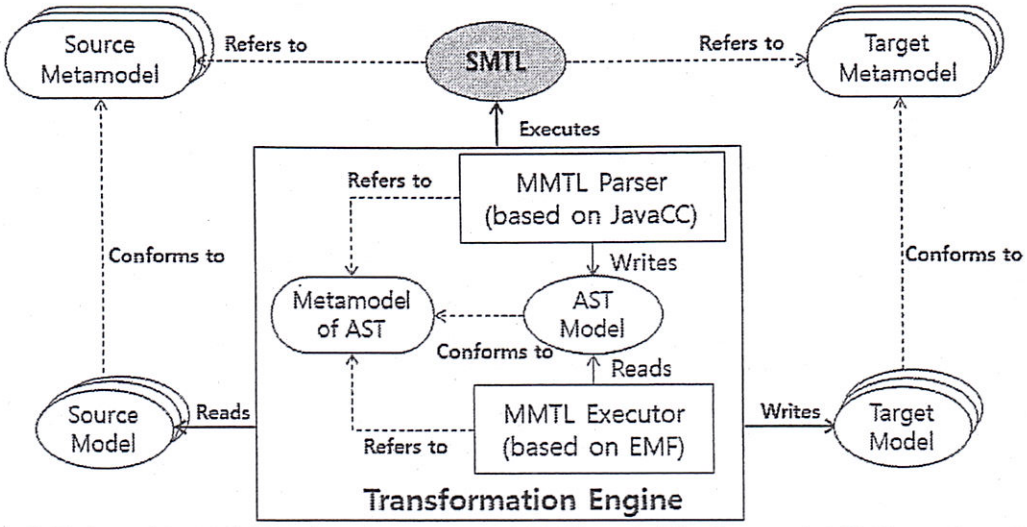
ATLAS Transformation Language (ATL) was a model transformation language and toolkit developed by ATLAS INRIA & LINA [11], which was based on metamodel and model transformation. In model-driven engineering (MDE), ATL provides to produce target models from source model developed on the Eclipse platform the ATL integrated Environment (IDE) provides standard development tools to easily develop ATL transformations[11].

ATL makes it possible to describe model transformations, which is a descriptive language of model transformation. ATL transformation program is composed of rules that define how source model elements are matched and navigated to create and initialize the elements of the target models. ATL language enables to define three kinds of ATL units: the ATL transformation modules, the ATL queries and the ATL libraries. According to their type, these different kinds of units may be composed of a combination of ATL helpers, attributes, matched and called rules. The ATL language is based on OMG OCL (Object Constraint Language) norm [17] for both its data types and its declarative expressions. But there exists a few differences between the OCL definition and the current ATL implementation.

In other words, ATL works that the class name of UML model inputted is transformed into the class name of Java code. C2C of Transformation rule is described about mapping definition of transforming Java class from UML class whether the class may be abstract class, pubic class, or one within any package. All source models Inputted into ATL should be UML models which be transformed into XMI models. For model transformation, it executes to transform from source model to Target model with ATL file defining Transformation rules and the produced file with the ATL File.

3. Development of Smartphone Model Transformation Language (SMTL)

SMTL transforms from source model to target model, which is based on metamodel like the previous model transformation method. SMTL is implemented with JavaCC [18] and EMF [19] like Figure 1. Transformation engine of SMTL is composed of Parser and Executor. Parser plays a role of making SMTL into abstract syntax tree (AST), and executor analyzes the AST based semantics, and executes metamodel transformation.



SMTL: Smartphone Model Transformation Language, EMF: Eclipse Modeling Framework, AST: Abstract Syntax Tree

Figure 1. Structure of Smartphone Model Transformation Language (SMTL)

Figure 2 shows an example of model transformation. To generate target model, it uses with "creating keyword". Also to update the data of source model, it uses with "updating and deleting keywords". This approach is possible to change both source and target models dislike the previous model. SMTL links elements of the particular metamodel. For example, UML class matches with Class Element of UML Metamodel.

```
IN(SOME_IN:UML2, SOME_IN2:UML2);
OUT(SOME_OUT:UML2);
Creating CreateClass from o:SOME_IN.Class {
      init {
            int num;
            string name;
            ref_list list1;
            ref_list list2;
            ref out = o.createClass;
            listAttr = o.getOwnedAttributes;
            listOper = o.getOwnedOperations;
            name = o.getName;
      }
      when { o.getName == "Class1" }
      do {
            out.setName = name;
            out.getOwnedAttributes.add = listAttr;
            out.getOwnedOperations.add = listOper;
      }
}
```

Figure 2. Example of SMTL

It executes with *init*, when, do to link each Element. *Init* defines a variable to execute transformation, and is possible to initiate data. When executes transformation only if a

value of the condition for transformation is true. *Do* is a part of generating actual model. To generate model, it does directly use EMF Application Program Interface (API).

Table 1 shows matching the relationship with SMTL and EMF API. For example, when *o.createClass* is executed, it transforms *o.createClass* into *createClass()*, and executes EMF API.

Table 1. The relationship between SMTL and EMF API

Type	SMTL	EMF API
Set data	setName = "name"	setName("name");
Get data	getName	getName()
Create element	createClass	createClass()
	createOperation	createOperation()
	createPropety	createPropety()
Get list	getOwnedAttributes	getOwnedAttributes()
	getOwnedOperations	getOwnedOperations()
Add element	getOwnedAttributes.add = element	getOwnedAttributes.add(element)
	getOwnedOperations.add = element	getOwnedOperations.add(element)

4. Case Study

In order to generate UML model based on metamodel, it can be substituted with some data of metamodel type. We separate with three types of *Data*, *Element*, *Element List*. Table 2 shows the possible combination of three types.

Table 2. Mapping data for model creation

		RHS		
		Data	Element	Element List
LHS	Data	Case 1	Case 2	Case 3
	Element	X	Case 4	Case 5
	Element List	X	X	Case 6

Case 1: Data to Data

Case 1 mentions all data values of both LHS and RHS. Like Figure 3, it uses character strings "*Class_After*" of RHS and *setName* of LHS to transform from the class name of Model A to the class of Model B. *o:UML.Class* behind '*from*' in "*Creating T1 from o:UML.Class*" is defined as the name o of the class element existed within UML metamodel.

The defined object '*o*' of metamodel is used as the basic element for transformation. SMTL consists of the directly linkable structure with method of EMF. Therefore, when SMTL is executed, *o.createClass* is mapping with method *createClass()*. That is, it defines a variable '*out*' with a keyword Ref within '*init*' for the produced object. '*out.setName="Class_After"*' is transformed into '*setName("Class_After")*'. This changes the name with "*Class_After*".

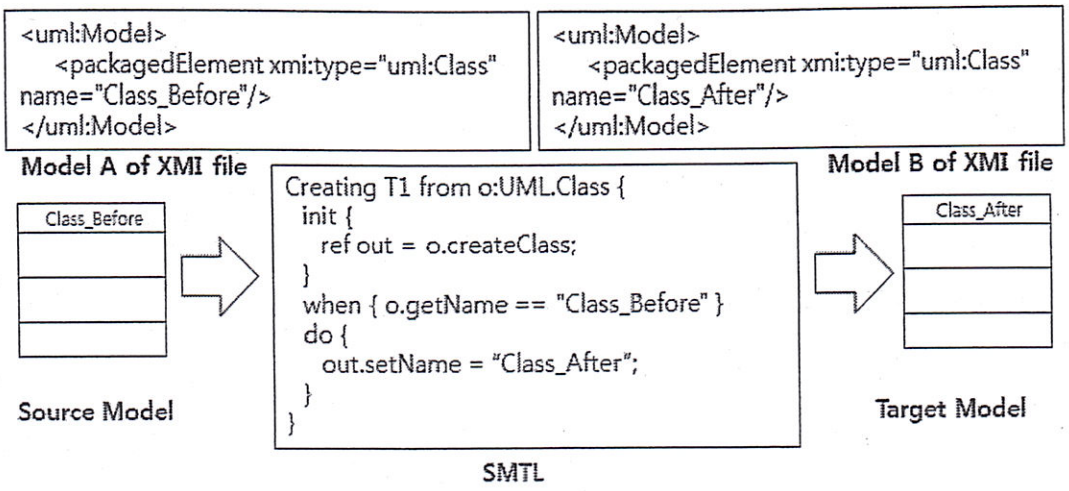


Figure 3. Mapping Data of Source model to Data of Target model

Case 2: Element to Data

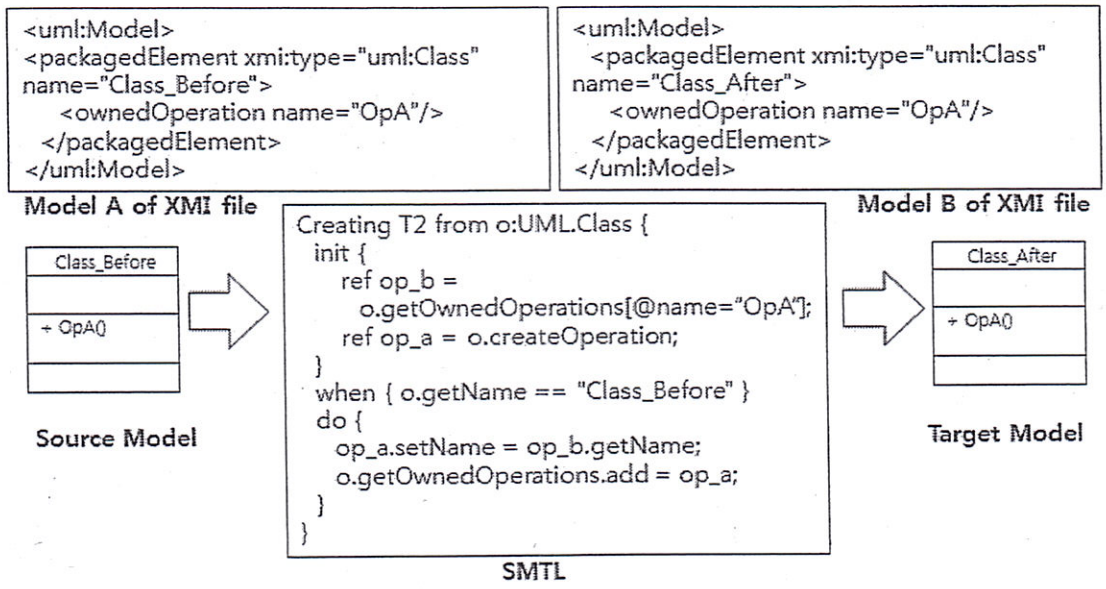


Figure 4. Mapping Element to Data

Case 2 mentions that LHS is *Data* and RHS is *Element*. When a method of Model A transfers the method of Model B like Figure 4, SMTL engine creates the method of Model B, and inputs the method's name of Model A from the name value of *Element*. This case is to create the method of target model from source model and to match method's name of both models. When invoking name value from Element, it selects a data in the multiple list data by using XPath expression. '*getOwnedOperation[@name="OpA"]*' selects an element named "*OpA*" in multiple methods. The SMTL engine is inputted the selected name of method element into target model

Case 3: Element List to Data

Case 3 mentions that LHS is *List* and RHS is *Data* in Figure 5. This case is to add a method in new class from the multiple elements in source model like Figure 5. In this case, a method of target model is created by the engine. To select an element between method lists of source model enters new method's name of target model. The engine inserts a newly created method in internal class by command named "*add*".

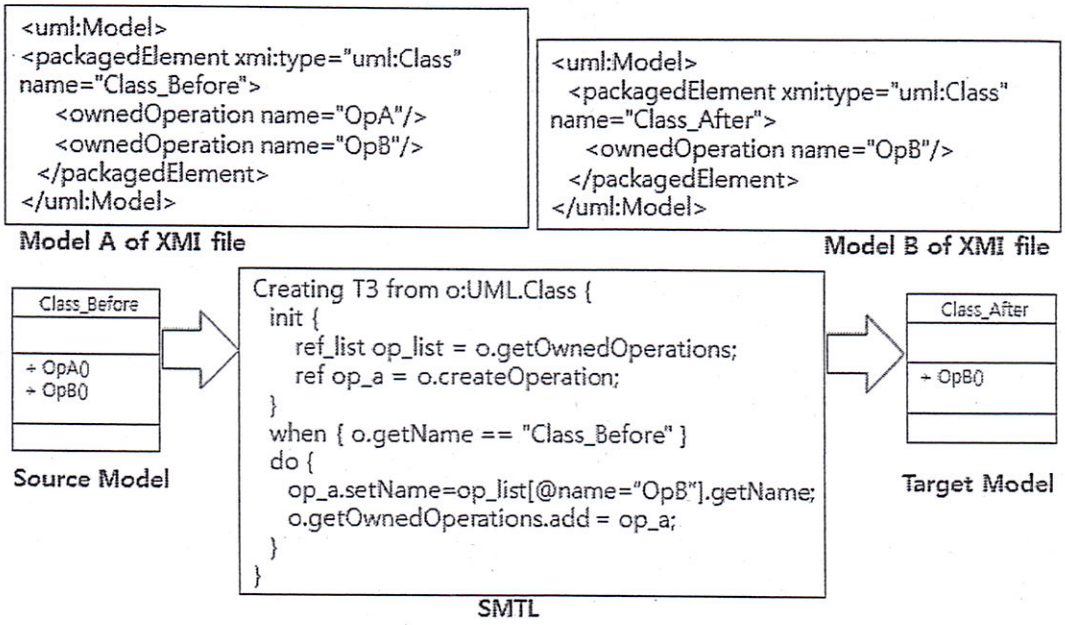


Figure 5. Mapping Element List to Data

Case 4: Element to Element

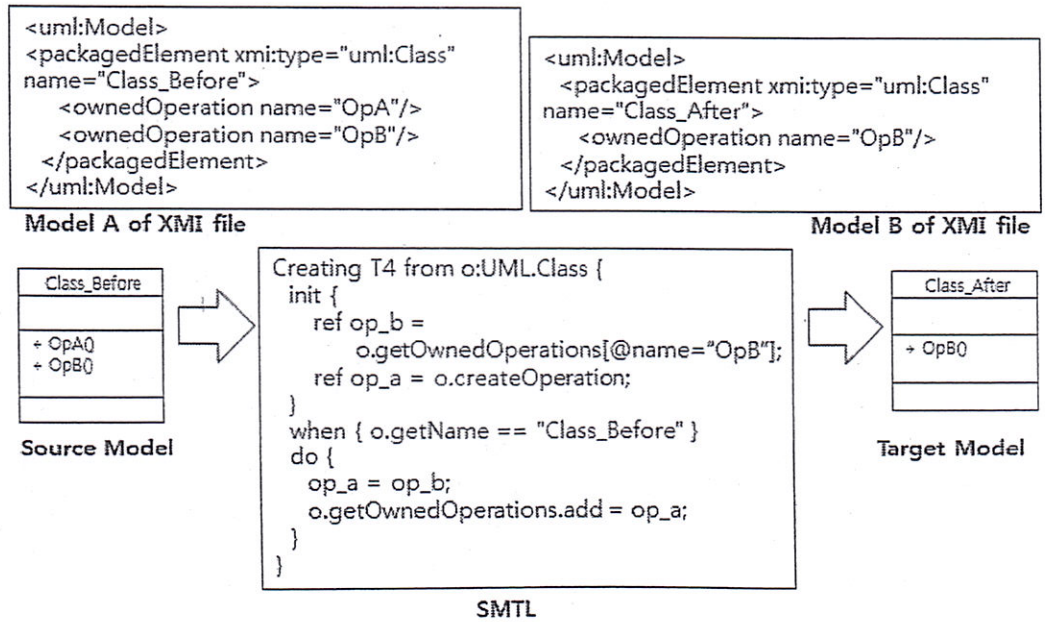


Figure 6. Mapping Element List to Element List

Case 4 mentions all elements of both LHS and RHS. This case transfers other element from an element in Figure 6. The SMTL engine adds method of the class in target model from selecting a method between two methods of source model.

Case 5: Element List to Element

Case 5 mentions that LHS is *List* and RHS is *element*. Like Figure 7, this case is to insert a selected method between method lists of source model into target model. Case 4 and case 5 look similar, but are distinct. The case 4 is to add the element that is defined element in *init*. The case 5 is directly selected the element in multiple list objects to add method.

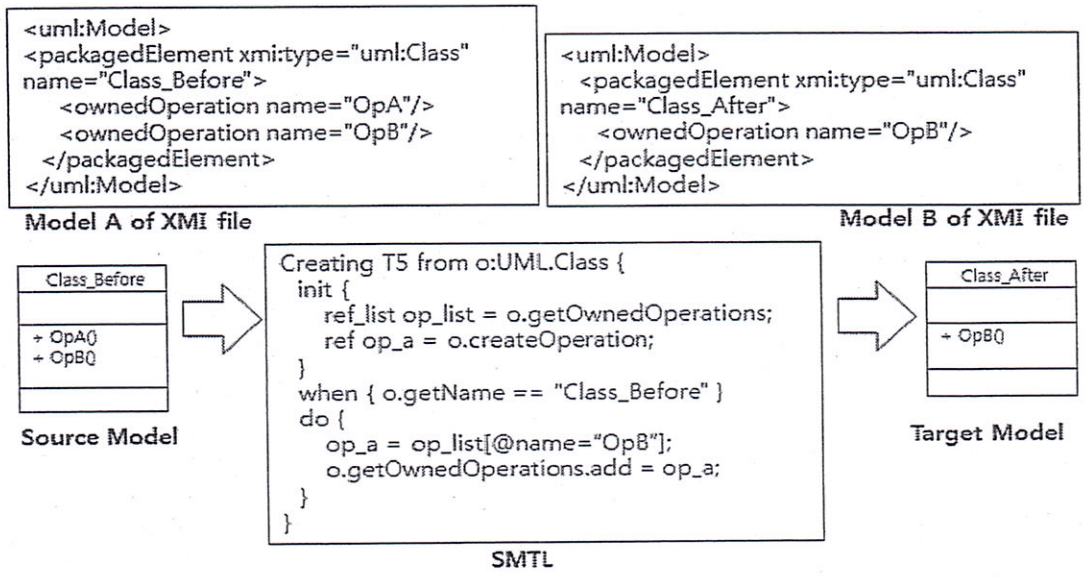


Figure 7. Mapping Data to Data

Case 6: Element List to Element List

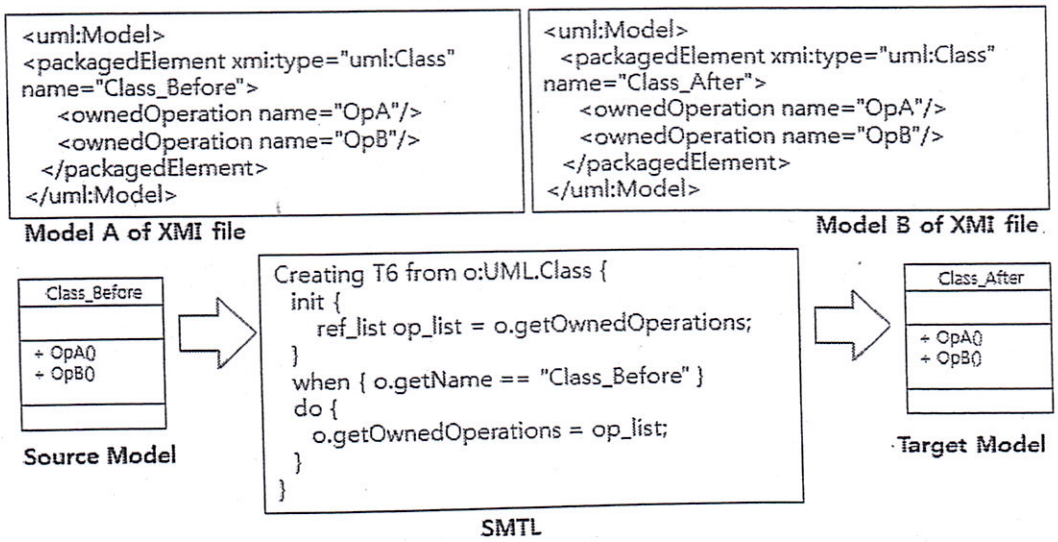


Figure 8. Mapping Data to Data

Case 6 mentions all element lists of both LHS and RHS. The methods of target model are copied with two methods in source model in Figure 8. When developer writes SMTL, this case is to add method using the assignment operator because two data types are the same.

5. Conclusion

Currently it does not exist to adapt smartphone area with model transformation. The model transformation through the existing ATL had limited with transformation rules and languages. So it was very complicated with transformation rules even though transforming a simple model. To solve this problem, we suggest SMTL model transformation language for heterogeneous smartphone development. Our suggested SMTL is designed to directly link EMF API for easily manipulate metamodel.

The proposed SMTL design to directly connect SMTL API with EMF API in order to manipulate easily the metamodel and to use XPath in order to search data in models. To retrieve data, it uses OCL, but not easy. So we apply to use XPath of XML. In order to do model transformation based on metamodel, SMTL should substitute data of multiple metamodel type. We separate with three types of *Data*, *Element*, *Element List* and the possible combinations are six cases. Then we made examples that separate six cases in order to demonstrate the ability of the SMTL and model transformation process. We can validate to cover the six cases from examples of the result through SMTL.

Further research will extend this work such as model transformation language and tools in order to use SMTL with heterogeneous smartphone environment in the future, which is not dealt in this study.

Acknowledgements

This work was supported by 2012 Hongik University Research Fund 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] Android, http://developer.android.com/.
[2] J. Yim, "Implementation of Building Recognition Android App.", International Journal of Multimedia and Ubiquitous Engineering, vol. 7, no. 2, **(2012)**, pp. 37-52.
[3] J. H. Yap, Yun-Hong Noh and Do-Un Jeong, "The Deployment of Novel Techniques for Mobile ECG Monitoring", International Journal of Smart Home, vol. 6, no. 4, **(2012)**, pp. 1-14.
[4] iPhone, https://developer.apple.com/.
[5] Windows Phone, http://dev.windowsphone.com/.
[6] D. Gavalas and D. Economou, "Development of Platforms for Mobile Applications: Status and Trends", Software, IEEE, vol. 28, no. 1, **(2011)**, pp. 77-86.
[7] W. Y. Kim, H. S. Son, J. S. Kim and R. Y. C. Kim, "Development of Windows Mobile Applications using Model Transformation techniques", Journal of KIISE: Computing Practices and Letters, vol. 16, no. 11, **(2010)**, pp. 1091-1095.
[8] W. Kim, H. Son, J. Yoo, Y. B. Park and R. Y. Kim, "A Study on Target Model Generation for Smartphone Applications using Model Transformation Technique", International Conference on Internet (ICONI) 2010, vol. 2, **(2010)**, pp. 557-558.
[9] B. Selic, "The pragmatics of model-driven development", Software, IEEE, vol. 20, no. 5, **(2003)**, pp. 19-25.
[10] K. Czarnecki and S. Helsen, "Feature-Based Survey of Model Transformation Approaches", IBM Systems Journal, vol. 45, no. 3, **(2006)**, pp. 621-645.
[11] Wikipedia, ATL, http://en.wikipedia.org/wiki/ATLAS_Transformation_Language.
[12] W. Y. Kim, H. S. Son, J. S. Kim and R. Y. C. Kim, "Adapting Model Transformation Approach for Android Smartphone Application", Advanced Communication and Networking, CCIS, vol. 199, **(2011)**, pp. 421-429.

[13] W. Y. Kim, "Model Transformation Framework for Heterogeneous Mobile Embedded Platforms", Hongik University thesis, **(2011)**.
[14] W. Y. Kim, H. S. Son and R. Y. C. Kim, "A Study on UML Model convergence Using Model Transformation Technique for Heterogeneous SmartPhone Application Software Engineering", Business Continuity, and Education, CCIS, vol. 257, **(2011)**, pp. 292-297.
[15] W. Y. Kim, H. S. Son and R. Y. C. Kim, "Design of Code Template for Automatic Code Generation of Heterogeneous Smartphone Application Advanced Communication and Networking", CCIS, vol. 199, **(2011)**, pp. 292-297.
[16] H. S. Son, W. Y. Kim, J. S. Kim and R. Y. C. Kim, "Concretization of UML Models based on Model Transformation for Windows Phone Application Information Science and Technology(IST 2012), **(2012)**, pp. 288-291.
[17] OMG, Meta Object Facility Specification, In OMG Unified Modeling Language Specification, Version 2.0 **(2006)**.
[18] T. Copeland, "Generating Parsers with JavaCC: An Easy-to-Use Guide for Developers", Centennial Books **(2009)**.
[19] D. Steinberg, F. Budinsky, E. Merks and M. Paternostro, EMF: eclipse modeling framework. Addison-Wesley **(2008)**.

Authors

Hyun Seung Son received his B.S. and M.S. degree in Software Engineering from Hongik University, Korea in 2009. He is currently a Ph.D. candidate in Hongik University. His research interests are in the areas of Automation Tool Development in Embedded Software, Real Time Operation System Development, Metamodel design, Model Transformation, and Model Verification & Validation Method.

Jae Seung Kim received his B.S. degree in Electronic Engineering from Hongik University, Korea in 2004. He is currently a M.S. candidate in Software Engineering from Hongik University. His research interests are in the areas of Automation Tool Development, Software Modeling & design, Metamodel design, Model Transformation, and Testing.

Robert Young Chul Kim received his B.S. degree in Computer Science from Hongik University, Korea in 1985, and the Ph.D. degree in Software Engineering from the department of Computer Science, Illinois Institute of Technology (IIT), USA in 2000. He is currently a professor in Hongik University. His research interests are in the areas of Test Maturity Model, Embedded Software Development Methodology, Model Based Testing, Metamodel, Business Process Model, and User Behavior Analysis Methodology.

International Journal of Software Engineering and Its Applications

IJSEIA

SCIENCE & ENGINEERING
RESEARCH SUPPORT SOCIETY