International Journal of Smart Home
Vol.8, No.3 (2014), pp.87-98
http://dx.doi.org/10.14257/ijsh.2014.8.3.08

Semantic-element-based Defining Approach for Model Transformation Rules

Lei Wang and Yuyan Zhang

School of Computer Engineering, Weifang University, Weifang, China

Abstract

In model-driven software development, it is a key technology that transform from platform independent models at higher abstract level to platform specific models at a lower level. The approach to create mapping rules is profoundly impacted with the gap between the source model and the target model. By abstractly analyzing the characteristic of syntax and semantics of modeling languages, an approach to define model transformation rules is proposed on the basis of semantic consistency. Firstly, the user must construct an abstract semantic model through an in-depth analysis of target platform. Secondly, the user build mapping relations from source model to target model via abstract target semantic model. This work is based on the idea of elements in source semantic domain being reconstructed in the target semantic domain. The approach can provide an effective support for validating mapping rules between different abstract level models. JavaEE is used as a target of the transformation to interpreting the process to define mapping rules.

Keywords: *Model-driven software development, Model transformation, Semantic consistency, Abstract level*

1. Introduction

Recently, model driven development becomes a hot topic and the main trend in software engineering, in which OMG's MDA [1] may be the most representative. There have been numerous research institutions and enterprises investing a large amount of money and manpower in the model transformation study. Currently, a number of products based on MDA have proved that a lot of benefits can be obtained from it, such as rapid development, architecture advantages, improvement of code consistency and maintainability, enhancement of system's portability across middleware vendors, and it also shows great potential in these areas [2].

On the whole, the provided approaches can be classified into five categories [3-6]: (1) Template-Based Approaches. In this approach, templates consisting of text in the target language include meta-code tags to access information from the source model. In the transformation process, these tags will be interpreted and eventually replaced by code representing the corresponding parts of the source. These approaches usually cover model-to-code generation. (2) Target-Structure-Driven Approaches. The basic metaphor is the idea of copying model elements from the source to the target. This kind of approaches was developed in the context of certain kinds of applications such as generating EJB implementations and database schemas from UML models. (3) Graph-Transformation-Based Approaches. This category of model transformation approaches draws on the theoretical work on graph transformation. In particular, these approaches operate on typed, attributed, labeled graphs, which is a kind of graphs specifically designed to represent UML-like models. This kind of approaches is inspired by heavily theoretical work in graph transformations, and it is powerful and declarative, but also the most complex ones. (4) Relational Approaches. This kind of approaches uses the mathematical concept of relations to specify how source and target models are linked. Relations are declarative but may be given execution semantics. It seems to strike a well balance between flexibility and declarative expression. (5) Transformation Implemented using XSLT. Models can be serialized as XML using the XMI, and implementing model transformations using XSLT.

Most of the approaches given above focus on providing a concrete solution for the transformation from platform independent models (PIMs) to platform specific models (PSMs), and there is little research on the definition principles for mapping rules as well as a basic theory to validate the mapping rules between such models. The research about natural language translation by machine shows that the prerequisite of correct transformation between different languages is the same or similar characteristics of semantics expression within the source and the target [7]. It is the same when talked about transformation between models at different abstract levels in MDA. A model mapping approach based on semantic consistency was proposed by abstractly analyzing the characteristic of syntax and semantics of modeling languages. Abstract target semantic model must be firstly constructed through an in-depth analysis of target platform. Then, based on the idea of elements in source semantic domain being reconstructed in the target semantic domain, mapping relations from source model to target model are created via abstract target semantic model. This approach may not only be a theoretical guidance for model transformation, but also be a measurement for validating the mapping rules between models at different abstract levels of the same system.

2. The Semantic Consistency Requirements of Model Transformation

2.1. Model Gap and Transformation

In MDA, a model is a representation of the function, structure and behavior of an application or system in a given formalism [8]. Any formalization language reflects a viewpoint that determines a set of modeling primitives and their semantics [9-11]. There is often a great difference between models at different abstract levels of the same system, and this situation is called isomeric features between different modeling descriptions in this paper. The isomeric features between different modeling descriptions are represented at three levels: syntax, semantics and structure. The syntactic gap refers to the difference among the date types and styles in different models. There also exist difference within the date structures, link ports and patterns of different models, and which called isomeric features at structure levels. The semantic gap means that the meanings of the terminologies used in different domains are not the same. The distance between them is more significant. The gap between the modeling languages can be narrowed using formalism extensions [11], but cannot be completely eliminated. The fundamental solutions for the problem of the gap seem to be the creation of effective semantics mapping mechanism, so to ensure that the equivalent representations for the system can be obtained [7, 12].

From operational view, a transformation is a terminating algorithm that applies structural and/or semantic changes to a model or a set of models. From function view, a transformation is a function that maps a tuple of models from one or more domains onto another tuple of models in the same or different domains [4]. Transformations are required not only to maintain semantic properties of the models but also maintain certain syntactical properties of the models.

2.2. The Semantics Consistency Relations between Different Models

Semantics is the meaning of information, which is relevant with its context. Some definitions are given bellow according to References [12] and [13]:

Definition 1: Semantic consistency refers to the case as follows: Let U and V be two different sets of elements, and APP be an application system, and then we take U and V as input to APP respectively. Two outputs named APP(U) and APP(V) were obtained respectively after the application's running over. The meaning and function of the two outputs are fully equivalence (or very similar), which is noted as APP(U)≌APP(V).

Definition 2: Semantic consistency of model mapping refers to the case as follows: Let MAP be a mapping from syntactic concepts to semantic domain, and when it applied to two concept patterns (named X and Y respectively) in different models, two set of primitives with equivalent semantics as the output can be obtained, which is noted as MAP(X)⇔ MAP(Y).

A consistency condition can be defined within the syntax expression based on a common semantic domain. In general, one distinguishes between two kinds of semantic consistency. Horizontal consistency problems exist for a set of models that describe the same aspect of a system from different points of view, potentially using different languages. It has to be ensured that these models do not contain contradictory concepts. Vertical consistency problems exist for models describing the same concept at different levels of abstraction. If a model is refined, it has to be guaranteed that the refined model does not contradict to the specifications of the more abstract model.

2.3. The Requirements for Semantic Consistency of Model Transformation in MDA

In model driven software development such as MDA, the source models are platform independent models of the system, and the target models (or target codes) are the further refinement with specific technologies based on certain platforms. The target codes will be converted into executable components after compiling. These components show target semantic model while they are running on target platform. The semantic consistency between source models and target semantic model is the fundamental requirement of model transformation, and it is also a basic measurement for judging the validity of mapping rules.

3. Model Mapping based on Semantic Consistency

The similar degree between models refers the size of the gap between the source and target models, in which syntax concept, organizational structure, semantic primitives and features will be considered. Its value may be varying in the range [0, 1]. The greater the value is, the higher similar degree will be.

3.1. Similar Degree between Models

As the source and the target models may be represented in different ways, it is hard to compute the similar degree between them directly. However, they both include many patterns in their respective description [14], so we can approximately compute the similar degree between models by using the definition of pattern matching.

Definition 3: A pattern is a combination of a set of conceptual variables and the relevant constraints which modeling elements bound to the pattern must satisfy [15].

A pattern can be defined as a 3-tuple: P = <C, A, SR>, where C= {c| c is a conceptual element in PIM or PSM}, and A= {a| a is a relevant attributes of the conceptual modeling elements}. Each attribute $a \in A$ is defined as a unitary relation a(c) in which $c \in C$ is the conceptual elements that a related to. SR = {kind-of, contain, associate ...} is the set of semantic relations between the conceptual modeling elements. Each semantic relation $sr \in SR$ is defined as a binary relation sr(c, c'), where c, c' $\in C$ and c relates to c' through sr. Thereby, the similar degree between models at different levels can be obtained approximately by using the matching degree of patterns involved in the models although some semantic information in the model will be lost (such as in constraints). The calculation process is rather simple and the lost information will not have severe impact on the result. The degree of pattern matching can be computed by adding the degree of concept matching and the degree of the context matching according to their weights. Concept matching degree shows the size of the gap between the meanings brought by the concepts, while context matching degree represents the similar degree about organizational structures and the relationship between the concepts [16].

3.2. Semantic Consistency based Model Mapping Approach

The size of the gap between the source and the target modeling language has a profound impact on the efforts to create mapping rules. The mapping relations are easy to define when the equivalent elements between the source and the target modeling languages can be determined. If the distance between two models is more significant, an intermediary model may be necessary to facilitate the mapping.

Under the guide of the semantic consistency principle given in Section 2, the approach used in this paper to define mapping relations is as follows: Firstly, abstract target semantic model must be constructed through an in-depth analysis of target platform within the limit of semantic constraints. Then the mapping relations from source models to the target semantic model and the mapping relations from the target semantic model to target models (or target codes) should be defined respectively. Thus, mapping relations from source model to target model can be built easily by taking abstract target semantic model as an intermediate, which is shown in Figure 1.

To construct abstract target semantic model, the relevant concepts should be gathered up by an abstract analysis, and then semantic information should be added to these concepts using constraints. In order to facilitate the automatic calculation of semantics, the conditions of constraints are restricted within the intersection of attributes, so to ensure the semantics transfer can be determined.

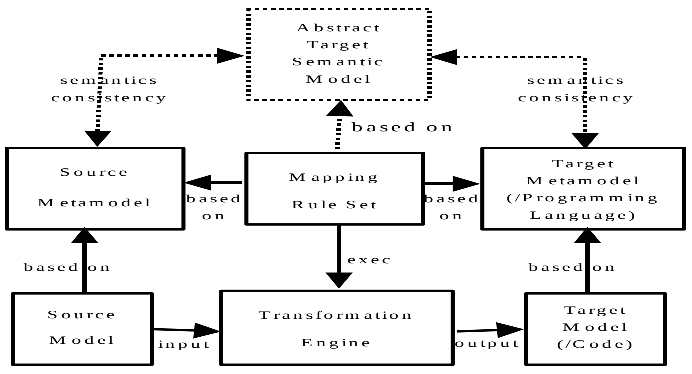


Figure 1. The Framework of Model Transformation based on Semantic Consistency

The semantics mapping between models can be considered as a reconstructing process in the target semantic domain for the elements in source semantic domain. That is to say, starting from source models, the values of relevant attributes can be obtained through observation and deduction on the source elements, and then to ascertain whether these values meet the requirements for the definition of target models and arrange accordingly [17].

Let C be the set of concepts of patterns in the target, *i.e.*, C = {c1, c2, ..., cn}. Let O1 be the set of attributes which can be observed directly, and O2 be the set of attributes observed implicitly, that is to say, these attributes of source models need to be deduced by using the context of concepts in the pattern, i.e. the necessary condition of these concepts, which is noted as {Nc| c∈C}. Let R be the classification rules for the attributes values of target semantic domain, *i.e.*, the sufficient condition for the classification in target semantic domain, which is noted as {Sc| c∈C}. Let M be the mapping relations between patterns. The mapping problems can be described as: to find a conceptual set ci in target semantic domain for each conceptual element cs in source domain, which satisfy the following equation.

$$O_1 \wedge O_2 \wedge R \Rightarrow M(c_s, c_i) \qquad (1).$$

The equation given above describes the mapping problem in formalism, i.e., a set of concepts of a pattern are given, which is noted as CS ={cS1, ..., cSm}. Then it can be mapped into target domain using the classification rules for the attributes values in target semantic domain noted as a concept set CT = {cT1 , ..., cTn}. The mapping process depends on semantics features in both ends. The source provides the observation of the source pattern (O= {NC| c∈CS}). The target provides target pattern and its classification rules (C= CT, R = {Sc| c∈CT}). By this way, a conceptual element of the source pattern can be mapped into target semantic domain through the process of finding the target conceptual set cTi which satisfy equation (1).The mapping relations from target semantic model to target models (or target codes) can be easily defined because it is very clear. We are no longer on it for the limited space.

4. A Case Study

The UML-based approach in [18] used as a source and JavaServer Faces assisted with Enterprise JavaBeans used as the target platform are shown in the case study to help interpreting the process of using the semantics consistency based model mapping approach.

4.1. A Modeling Approach for Platform Independent Models

The modeling approach proposed in [18] is based on extending UML and introduces user-interface presentation views. In this approach, there is an abstract description of UI component data and behavior elements rather than a list of interface elements and their attributes. At the same time, the binding relations between UI elements and the corresponding objects are given, which made both the data objects and the behavior elements be independent on concrete UI components and widgets. The FMP can be used to build platform independent models for Web applications as the source in model transformation. Its contents are composed of two layers: architecture modeling and component modeling.

System represents the architecture and constraints of a software system, which is defined as a 4-tuple: <Style, Description, ComponentSet, Relations>. Style represents the architecture style. Description represents functional descriptions for the system. ComponentSet represents the set of components and connectors. Relations is a list of relations among components and connectors [19].Component is the foundation of software system for function design and realization.

Function View, Workflow View, Static View, Action View and UI Presentation View are used in the FMP approach to build component models. Each view represents an aspect of the application system.

The functions of components in architecture model, exchange information between the system and outside, and the interactions among function modules of the system are all described using Function Views. It uses Use-Case Diagram in UML to complete description, which is defined as a 3-tuple: <RoleSet, UCSet, AssocSet>. UCSet is the set of use cases, and it is used to describe system's function. RoleSet represents the set of roles, which is used to describe the user of use cases. AssocSet is a set composed of the using relations between roles and use cases.

Workflow Views are used to model the actions of each individual, and define interactive relations and cooperative relations among these entities. It uses state-machine based activity diagrams to complete description, which is defined as a 4-tuple: <InitState, ActivitySet, CondiSet, FinalStateSet>. InitState is the state-machine's initial state. ActivitySet is a set of activities. CondiSet represents the set composed of change conditions for states. FinalSateSet is a set of final states of the state-machine.

Static View is an integration of Package Diagram and Class Diagram in UML. It is used to describe analytical classes of the use cases in Function View and the relations among these classes. Static View also includes the information about structure features of a sub-system, and it is defined as a 3-tuple: <ClassSet, PackageSet, AssociSet>. ClassSet is the set of classes. PackageSet is the set of packages. AssociSet is a set including the relations among classes and the relations among packages, and it also includes the relations between classes and packages.

Action View uses extended Collaborative Diagram in UML to describe the actions of objects in more detail. It is defined as a 4-tuple: <RoleSet, ANSet, ObjectSet, AssociSet>. RoleSet is the set of roles. ANSet is a set composed of Action-Nodes, which are abstract representation symbols for the connecting points of system action. The association from Role to Action-Node represents the using relations between them. ObjectSet is the set of objects, and AssociSet is the set of relations between these modeling elements.

Action-Nodes are represented by the ellipses. Object is represented as a rectangle, and Data Collection is represented with 2 overlapped rectangles. Data Object and Data Collection have the property of their data source, which is shown as an additive cylinder in rectangle. Rectangle with two vertical bars is the symbol for other UI Presentation Views. Dotted arrow directed to it means the relations of UI navigation. Rectangle with one or more small circles connected to it represents an external entity or component, where the circles are its Entry Points. Most of the symbols have the property of Visible or Non-Visible on UI. Visible object is represented as a rectangle with real line. Non-Visible objects are represented as rectangles with dashed line.

The implication of Action View is as follows: the role uses the system's function by touching off Action-Node, thus arose message to transmit along these objects. Next UI page is selected according to the results after the function execution was completed.

UI Presentation View provides intuitional presentation for boundary objects and the interaction points between the users and system in Action View. It also provides binding relations between UI modeling elements and the visible objects in Action View. UI Presentation View is defined as a 2-tuple: <AreaNodeSet, LayOutStrategy>, AreaNode= <UIComponentSet, UCActionSet, UCLayout>. A presentation page is divided into several presentation areas (AreaNode), and each area has a layout strategy (LayOutStrategy). An area also can be divided to several sub-areas. UIComponentSet is a set composed of UI presentation components, such as Data-grid, Form, Graphics, *etc*. UCActionSet represents the set of interaction points in Action View corresponding to the presentation area.

4.2. JSF+EJB: the Target of Model Mapping

JavaServer Faces (JSF) [20] is a new standard Java framework for building Web applications, which developed through Java Community Process (JCP). It simplifies development by providing a component-centric approach to developing Java Web user interfaces. JSF also ensures that applications are well designed with greater maintainability by integrating the well-established Model-View-Controller (MVC) design pattern into its architecture. This makes JSF applications much more manageable because the user-interface code (View) is cleanly separated from the application data and logic (Model).

JavaServer Faces assisted with Enterprise JavaBeans (EJB) make a good balance between the efficiency of systems development and the costs for system maintenance, which can be used to develop comprehensive web applications supporting various data types or clients (such as HTML browser and WML browser) to meet the requirements for stringent safety and transaction processing. In this paper, JSF+EJB are used as the target platform for model transformation.

As shown in Figure 2, an abstract target semantic model for JSF+EJB was defined based on the MVC design pattern, and its components are divided into three kinds: Static Component, Action Component and Presentation Component.

The Model Layer (Static Component) contains detail semantic information about Java application programs and EJB specification, such as Package, Java-Interface, Java-Class, Attribute, Method and the relationships between Java-Classes.

The Controller Layer (Action Component) is used to describe a system from dynamic aspect, and its elements are organized by surrounding the solution of system tasks. According to the solution process of user’s request, action component model is constructed with reference to the interaction relations between users and the system.

There are two kinds of action elements (WebAction) in Action Component model. The first kind represents the entry-points for interactions between users and the system, and it can be touched off directly. The second kind of WebAction represents action elements within the system and which can be touched off via the first kind. Navigation is the target of next step after the request is resolved. ActionPara represents the parameter object applied in the solution process of a system action. DataObject represents the kind of object that is the target of an operating. Invoke is an invoked relation from WebAction to DataObject.

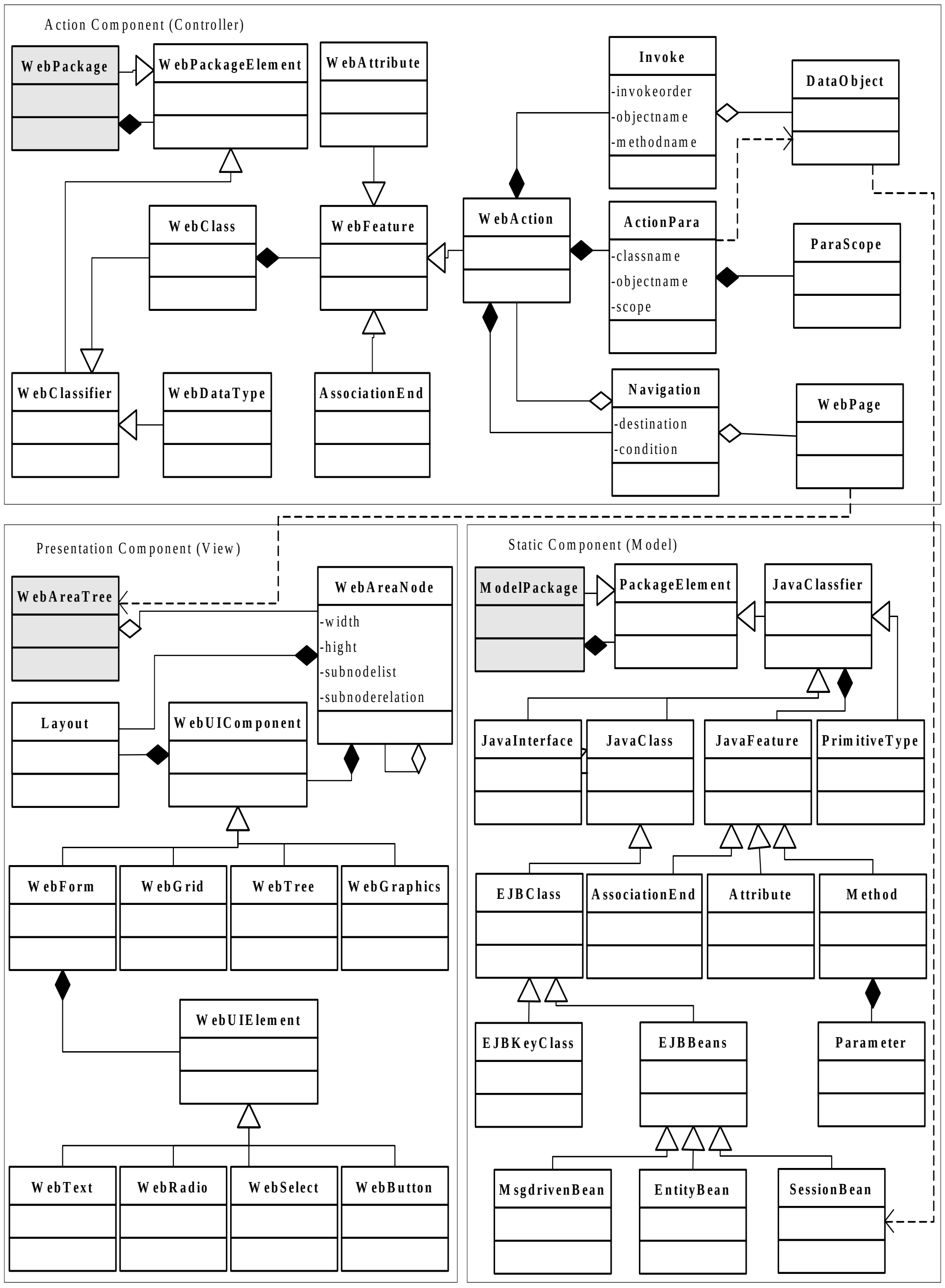


Figure 2. A Sketch of Abstract Target Semantic Model based on JSF+EJB

The essential semantic information brought by Action Component model is as follows: users touch an WebAction, and the application system receives user’s request that maybe including ActionParas, then it analyses and dispatches the request to the corresponding actions. WebAction invokes the method of Invoke-Objects to resolve the task. After completing, actions forward the request to next page according to the result and the conditions of Navigations.

The View Layer (Presentation Component) is organized as a hierarchical tree-like form to represent the specific relationships of UI elements for Web applications. Each UI page is represented as a WebAreaTree, which contains several WebAreaNodes and a layout strategy (Layout). Each WebAreaNode maybe include some WebUIComponents, such as WebForm, WebGrid, WebTree, *etc*.

4.3. The Mapping Relations

According to the semantic consistency based modeling approach presented in Section 3 and taking the abstract target semantic model for JSF+EJB (JSFATSM) as an intermediary, we define mapping relations form source model (PIM) to target model (JSFTM) according to the syntax and semantic features of modeling elements. Complex rules can be constructed by simple mapping rules. The holistic mapping relations are shown in Figure 8: Entity-objects in PIM's Static View are mapped to Entity-beans of JSFATSM. Control-objects are mapped to Session-beans. Boundary-objects are mapped to ActionPara or DataObject in Action Component and UI presentation elements in Presentation Component. The information brought by UI Presentation View should be mapped into the Presentation Component of JSFATSM.

The mapping relations from abstract target semantic model to target models are more obvious and easy to build. The main work is the analysis, restructuring and integration for the information within target semantic model, and which also includes the addition of the corresponding information about target platform. The Static Component model in JSFATSM are mapped to the corresponding EJB components. The information within Action Component model are mapped into the business logic module, navigation processing module and the mapping relations of the configuration files. The information within each WebAreaTree in UI presentation model mapped into the corresponding active server page files, which are mainly for UI layout, presentation components and UI widgets.

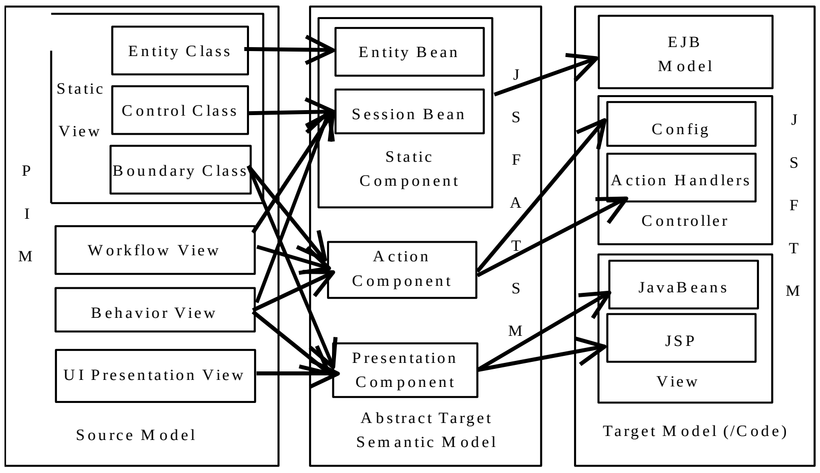
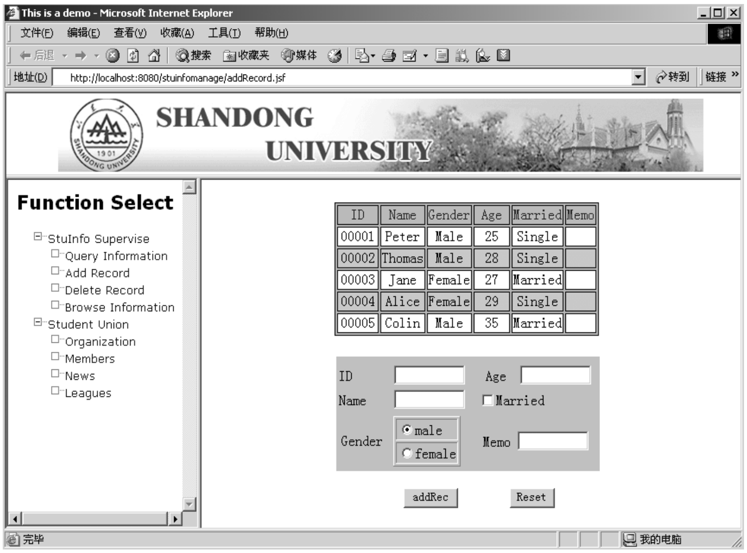


Figure 3. The Mapping Relations from PIM to JSFTM via ATSM

As can be seen from Figure 3, the introduction of abstract target semantic model simplifies the definition of mapping relations from PIM to PSM. Template-Based Approach [3] can be used to realize the generation of target codes, which is widely used in MDA-supported tools, such as AndroMDA, OptimalJ, ArcStyle, and which is not repeated in this paper. The static view, the action view and the UI presentation view are all integral parts of a PIM model for a student information management system. After two steps of transformation (from PIM to PSM, from PSM to target codes), the actual running page based on JSF framework is shown in Figure 4.



ID	Name	Gender	Age	Married	Memo
00001	Peter	Male	25	Single	
00002	Thomas	Male	28	Single	
00003	Jane	Female	27	Married	
00004	Alice	Female	29	Single	
00005	Colin	Male	35	Married	

ID
Age
Name
Married
Gender [✓ male] female
Memo

Figure 4. An Actual Running Page

5. Conclusion and Future Work

Starting from the analysis of semantic consistency requirements of model transformation, a model mapping approach based on semantic consistency was proposed in this paper. Based on the idea of elements in source semantic domain being reconstructed in the target semantic domain, this approach can be used to build mapping relations from source model to target model. Target semantic model is considered as a reference for disambiguation, and which can provide a good basis for the semantic comparison between modeling languages at different abstract levels (such as UML and target codes). Therefore, by using this approach, semantic consistency between different descriptions of the same component can be ensured. At the same time, the model transformation process is accompanied with a process of model validating, which can provide an effective support for model driven development.

Future works are as follows: (1) more study about formal description of target semantic models, and thus to strengthen the abilities of semantic expressiveness and consistent verification between models; (2) further formalize the model mapping process for the enhancement of accuracy; (3) to completely abstract the description related to UI presentation in target semantic model, and enhance visual attractiveness of the generated page; (4) to diversify target platform in order to verify the practicability of this approach.

Acknowledgments

The authors are most grateful to the anonymous referees for their constructive and helpful comments on the earlier version of the manuscript that helped to improve the presentation of the paper considerably. This research is supported by the Foundation of Science-technology Development Project of Shandong Province of China under Grant No. 2011YD01042.

References

[1] M. Völter, T. Stahl, J. Bettin and A Haase, “Model-Driven Software Development: Technology”, Engineering, Management, Jojn Wiley &Sons, Inc. **(2006)**.

[2] I. Malavolta, H. Muccini, P. Pelliccione and D. A. Tamburri, “Providing architectural languages and tools interoperability through model transformation technologies”, IEEE Trans. on Software Engineering, vol. 36, no. 1, **(2010)**, pp. 119-140.

[3] K. O. Elish and M. Alshayeb, “Using Software Quality Attributes to Classify Refactoring to Patterns”, Journal of Software, vol. 7, no. 2, **(2012)**, pp. 408-419.

[4] J. Hou, “Mathematical Description Framework for Architecture Models”, Proc. of the Second International Workshop on Education Technology and Computer Science, pp. 149-152.

[5] V. Dahi and D. P. Saint, “Natural Language Understanding and Logic Programming”, Elsevier Science Publishers, **(1985)**.

[6] I. Traore, I. Woungang, A. A. E. Sayed Ahmed and M. S. Obaidat, “Performance Analysis of Distributed Software Systems: A Model-Driven Approach”, Proc. of IEEE Intl. Symposium on Performance Evaluation of Computer Telecommunication Systems (SPECTS 2010), **(2010)**, pp. 111-118.

[7] J. Yu, H. Cai and F. Bu, “OWL-S based Service Composition of Threedimensional Geometry Modeling”, Journal of Multimedia, vol. 7, no. 1, **(2012)**, pp. 98-105.

[8] J. Kwon, D. Jeong and L. S. Lee, “Intelligent Semantic Concept Mapping for Semantic Query Rewriting/ optimization in Ontology-Based Information Integration System”, International Journal of Software Engineering and Knowledge Engineering, vol. 14, no. 5, **(2004)**, pp. 519-542.

[9] S. Beydeda, M. Book and V. Gruhn, “Model-Driven Software Development-Volume II of Research and Practice in Software Engineering”, Springer, **(2005)**.

[10] M. Li, L. Yuan and W. Du. “An Adaptive Motion Model and Multi-feature Cues Based on Particle Filter for Object Tracking”, Journal of Multimedia, vol. 7, no. 5, **(2012)**, pp. 364-371.

[11] S. Philippi, “Automatic code generation from high level Petri Nets for model driven systems engineering”, Journal of Systems and Software, vol. 79, no. 10, **(2006)**, pp. 1444-1455.

[12] A. Filieri, C. Ghezzi, A. Leva and M. Maggio, “Self-Adaptive software meets control theory: A preliminary approach supporting reliability requirements”, Proc. of the 26th IEEE/ACM Int’l Conf. on Automated Software Engineering, **(2011)**, pp. 283-292.

[13] T. Jiang and X. Wang, “Formalizing Domain-Specific Metamodeling Language XMML Based on First-order Logic”, Journal of Software, vol. 7, no. 6, **(2012)**, pp. 1321-1328.

[14] M. Stefik, “Introduction to Knowledge Systems”, Morgan Kaufmann Publisher, **(1995)**.

[15] M. Bernardo, P. Ciancarini and L. Donatiello, “Architecting families of software systems with process algebras”, ACM Trans. on Software Engineering and Methodology, vol. 11, no. 4, **(2002)**, pp. 386-426.

[16] R. Lu, “Towards a Mathematical Theory of Knowledge”, Journal of Computer Science and Technology, vol. 20, no. 6, **(2005)**, pp. 751-757.

[17] I. Traore, I. Woungang, A. A. E. Ahmed and M. S. Obaidat, “Software Performance Modeling using the UML: a Case Study”, Journal of Networks, vol. 7, no. 1, **(2012)**, pp. 4-20.

[18] J. Hou, J. Wan and X. Yang, “MDA-based Modeling and Transformation Approach for WEB Applications”, Proceedings of the sixth international conference on Intelligent System Design and Applications (ISDA'06), Jinan, Shandong, China, **(2006)**, pp. 867-812.

[19] T. Mens, N. E. Van, S. Demeyer and D. Janssens, “Formalizing refactorings with graph transformations”, Journal of Software Maintenance and Evolution: Research and Practice, vol. 17, no. 4, **(2005)**, pp. 247-276.

[20] D. Varro, “Automated formal verification of visual modeling language by model checking”, Journal of Software and Systems Modeling, vol. 3, no. 2, **(2004)**, pp. 85-113.

[21] I. Weisemöller, F. Klar and A. Schürr, “Development of tool extensions with MOFLON”, Lecture Notes in Computer Science, vol. 6100, **(2011)**, pp. 337-343.

[22] Y. Liu, Y. Gu and C. Jun, “A New Control Structure Model Based on Object-oriented Petri Nets”, Journal of Networks, vol. 7, no. 4, **(2012)**, pp. 746-753.

Author

Lei Wang He is currently working at Weifang College as a lecturer. He received his M.S. degree in the school of Software at the Shandong University, China, in 2006 and his Ph.D. degree in the School of Computer Science and Technology at the Shandong University, China, in 2010. His research interests are in the areas of graphics, vision and human-computer interaction.. He is a member of CCF.