International Journal of Future Generation Communication and Networking
Vol. 9, No. 12 (2016), pp. 329-338
http://dx.doi.org/10.14257/ijfgcn.2016.9.12.30

Developmental Approaches Covering Context Area Mobile Applications Service Oriented Architecture and Model Driven Architecture

Haeng-Kon Kim[1], Hyun Yeo[2], Tai-hoon Kim[3], Carlos Ramos[4], Goreti Marreiros[5] and Ha Jin Hwang[6]

[1]*Department of Computer Engineering, Catholic University of Daegu, Korea*
[2]*Department of Computer and Communications, Sun Cheon National University, Korea*
[3]*Department of Convergence Security, Sung Shin Women's University, Seoul, Korea*
[4,5]*Institute of Engineering, Polytechnic of Porto, Porto, Portugal*
[6]*Department of Business Analytics, Sunway University Business School, Malaysia*
hangkon@cu.ac.kr[1], yhyun@sunchon.ac.kr[2], taihoon@daum.net[3], csr@sc.ipp.pt[4], mgt@isep.ipp.pt[5], hjwang@sunway.edu.my[6]

Abstract

Model Driven Architecture (MDA), as we consider the growing significance and utility of modeling in the development of software and solutions, it reflects the benefits of MDA to transform one PIM into several PSMs, each for platform or technology in which the final system will be deployed, and the automatic code generation that implements the system for those platforms from the corresponding PSMs. Service-oriented architectures (SOA) are also presented as the key to business agility, especially when combined with a model-driven approach. Model-Driven Architecture (MDA) is a well-developed idea that fits well with SOA, but as of today, it has been a specialized technique that is beyond practical application scope of most enterprises. We describe the MDA and SOA abstract components to be useful in mobile business applications in the future, allowing to add the features of the two modeling architectures, concentrating on the classification of models that is embodied individually. The framework given, a unified modeling architecture, which illustrates how the two architectures can be brought together into one.

Keywords: *Model-Driven Architecture (MDA), Context Aware Mobile Applications Domain Model, Service-oriented architectures (SOA), Software Process Improvement, Component Based Development*

1. Introduction

Developing and designing a system requires a lot of effort, but the object-oriented and component-based technology has not met the needs of these systems yet, and may be considered adding some new ideas that need simplification. [1-4]. Service-oriented architecture (SOA) this is to loosely coupled, protocol independent, standards-based distributed computing where software resources available on the network are considered as Services [3]. The service-oriented architecture (SOA) approach and the web service standards such as the Web Service Description Language (WSDL) [5] and the Simple Object Access Protocol (SOAP) [6] are currently adopted in various fields of distributed application. As model driven architecture (MDA) [7]. has been provided as an approach to work with complex software systems by dividing the development process into 3separate model:

1) Platform Independent Model (PIM) layer holds a high level representation of the

ISSN: 2233-7857 IJFGCN

entire system without committing to any specific operating system, middleware or programming language. It PIM provides a formal explanation of an application's functionality without burdening the user with too much detail.

2) Platform Specific Model (PSM) layer holds a representation of the software specific to a certain target platform such as J2EE, Corba or in our case the service oriented Grid middleware.

3) Code Layer consists of the exact source code and supporting files which can be compiled into a working piece of software. Every specific part of the system is completely specified. MDA theory explains that a PIM is specified and automatically changed into a PSM and then into actual code, which makes the system design easier to use. The main idea lies in the development of generic transformers which is capable of generating PSM and code layers from the PIM [8]. Mobile business application on SOA is a relatively young field of distributed computing and not complete in any form of tool to support a model driven approach in development a software. This is a negative view since we believe that due to its high complexity/accuracy, and the high rate of churn in the software technology market, MDA approach is vital to the adoption of this new technology. Only if "business logic" (i.e. application functionality) developers can more or less effortlessly integrate a new middleware into their system, will a widespread adoption be possible. The developers who are responsible for the integration of the middleware into the overall system should concentrate on middleware concerns and not have to cope with the business logic as well. By using an appropriate MDA approach, this division of concerns can be facilitated accordingly.

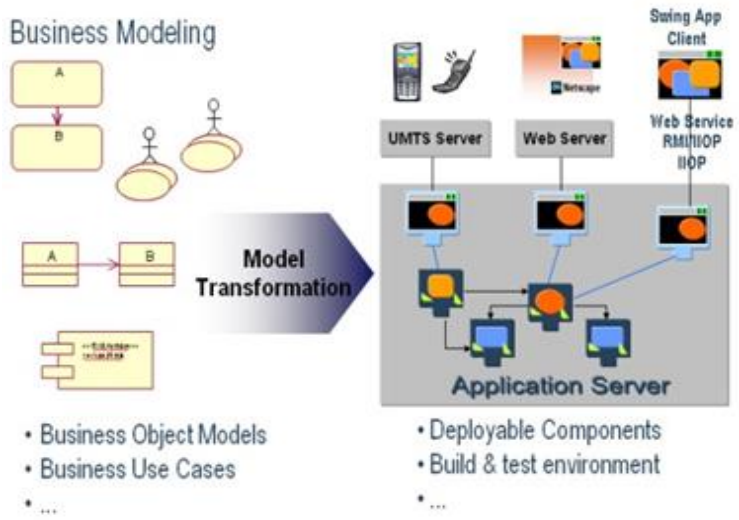

Figure 1. MDA Approaches

In this paper, we present presented a model-driven approach to SOA modeling and designing complex distributed systems based on MDA. MDA separates the Platform Independent Model (PIM) from the Platform Specified Model (PSM) of the system and transforming between these models via appropriate tools. The PIM of the system is build and then the PSM based on SOA is generated. The final PSM based on a target platform is generated. These models are generated with transformation tools in MDA and an approach to the model driven development with mobile business applications on SOA is presented. This is to minimize the necessary human interaction required to transform a PIM into a PSM and a PSM into code for a SOA. To separate the architectures specific components from the business specific components, a UML mobile business profile is introduced and a separation of the PSM layer into two parts which make the automated transformations from PIM to PSM to code easier to implement. The separation introduced

on the PSM layer is reflected on the code layer by the use of Java annotations.

2. Related Works

2.1. Modeling Mobile Services Metadata based on MDA

This present a perfect framework for application-to-application integration or collaboration, to make it possible for these applications to be available as Web services as mobile services are emerging.

To systematize the use of Web services, the World Wide Web Consortium (W3C) proposed the Web Service Description Language (WSDL) standard, an XML-based language that describes the Web service roles. Basically a WSDL file is a language-independent XML-based version of an IDL (Interface Definition Language) file that defines the operations offered by a Web service, same as the parameters that these operations accept and return. So, WSDL made its way to be the standard that supports the description of Web services: What they do, how they should be used, and where they are localized [8-9].

The WS-Policy framework consists of two specifications: WS-Policy and WS-Policy Attachment.

- **WS-Policy** defines the syntax for expressing policy alternatives and for composing them as combinations of domain assertions. The WS-Policy specification also describes the basic mechanisms for merging multiple policies that apply to a common subject and the intersection of policies to determine compatibility.
- **WS-Policy Attachment** defines how to associate policies with a particular subject. It provides normative descriptions of how this applies in the context of WSDL and UDDI, (Universal Description, Discovery, and Integration), it offers an extensible mechanism for associating policies with arbitrary subjects through the expression of scopes.

As Mobile services and the thriving Model Driven Architecture (MDA) made it's ay to fame, we must consider the rising significance and utility of modeling in the development of software and solutions. MDA, which was proposed by the Object Management Group (OMG), is a model-driven framework for software development. Listed advantages of MDA are the ability to transform one PIM into several PSMs, 1 for each platform or technology in which the last step of the system, and the automatic code generation that implements the system for those platforms from the corresponding PSMs.

As Mobile services are software components, the development of Mobile services must exploit the advantages of MDA. To apply the MDA principles in the development of Web services, a modeling process must be considered. According to MDA principles, this modeling activity should result in automatic code generation. If we want to abstract from the platform in the Web service that will be deployed, the code to be be generated is the WSDL document that contains the Web service description in a standard format.

2.2. MDA Main Concepts

The main concepts of the MDA are beginning to be identified [6-7]

MDA stands for Model Driven Architecture. It is a framework for software development by OMG. Within MDA the software development process is driven by the activity of modeling your software system. A meta-model acts as a filter to extract some relevant aspects of a system and to ignore all other details. A meta-meta-model defines a language to write meta-models. There are several possibilities to define a meta-meta-model. Usually the definition is reflexive, i.e. the meta-meta-model is self defined. A meta-meta-model is based at least on three concepts (entity, association, package) and a set of primitive types. It contains all universal features (The OMG MOF), *i.e.* all those that are not specific to a particular domain language. Among those features we find all that is

necessary to build meta-models and to operate on them. Maintaining a specific tool for the MOF would be costly, so the MOF is aligned on the CORE part of one of its specific metamodels: UML. UML thus plays a privileged role in the MDA architecture. As a consequence, any tool intended to create UML models can easily be adapted to create MOF meta-models. MDA uses models and a generalized idea of architecture standards to address integration of enterprise systems in the face of heterogeneous and evolving technology and business domains. MDA combines computer-aided verification and machine intelligence during modeling to discover and remove design bugs before code reviews and testing. It acts as a filter, the MDA Meta model that extracts some relevant aspects from a system and to ignore for all other details. A meta-meta-model defines a language to write meta-models. The application of MDA to a use case begins by focusing on the development of the models. Figure 2 below, show the MDA process as follows: Computation Independent Model (CIM): describes concepts of a given domain but does not describe the software system. Platform Independent Model (PIM): describes software behavior that is independent of some platform. Platform Specific Model (PSM): describes software behavior that is specific for some platform. The initial step in using MDA is to make and improve a CIM which describes the concepts for a specific domain. The CIM focuses on the environment and the requirements of the system; the details of the structure and processing of the system are hidden or as yet undetermined. The next step involves developing the PIM. The term "platform" can have various meanings and can include one or more system aspects such as operating system, programming language and network configurations, PIM and PSM are models are therefore relative to the definition of platform used in the use case. More important than the definition of platform is the recognition that PIMs and PSMs are supposed to separate aspects of program behavior from aspects of implementation. The third step is developing one or more PSMs which characterize a particular deployment of a software application. This could, for example, focus on the properties of a web application, whether the application should be generated in Java or Visual Basic, or whether the installation was for a standalone or networked machine. MDA requires the development of explicit transformations that can be used by software tools to convert a more abstract model into a more concrete one. A PIM should be created, and then transformed into one or more PSMs, which then are transformed into code." The mappings between models are meant to be expressed by a series of transformation rules expressed in a formal modeling language. "A CIM is a software independent model used to identify a business system. Certain parts of a CIM may be supported by software systems, but the CIM itself remains software independent. Automatic derivation of PIMs from a CIM is not possible, because the choices of what pieces of a CIM are to be supported by a software system are always human. For each system supporting part of a CIM, a PIM needs to be developed first."

It is possible for concepts defined in a CIM to be automatically associated with properties defined in a PIM. For example, the concept "protein" defined in a CIM about proteomics experiments could be associated with PIM concepts such as a help feature that defined protein for users or a drop down list of protein names.

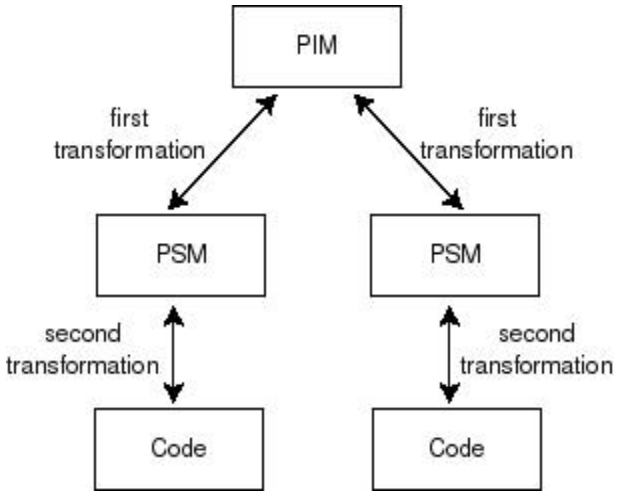

Figure 2. MDA Development Process

A meta-model in MDA defines a specific domain language. This could be compared to the formal grammar of a programming language. In UML, the requirement to define variants of the base language was expressed. The UML meta-model was then equipped with extension mechanisms such as stereotypes, tagged values, constraints and this allows defining specialization of the basic meta-models as so called profiles.

The MOF includes features to serialize models and meta-models in order to provide a standard external representation. The XMI standard defines the way serialization is performed. This is a way to exchange models between geographical locations, humans, computers or tools. When a tool reads a XMI serialized model (a UML model for example), it needs to check the version of the meta-model used and also the version of the XMI applied scheme.

2.3. SOA

Service-Oriented Architecture (SOA) is a software design where services are provided to the other components. This services communicate with each other. It exposes real dependencies against artificial ones [11]. A real dependency is a state of affairs in which one system depends on the functionality provided by another. Beside real dependencies there are always artificial dependencies in which the system becomes dependent to configurations and various requirements of other systems expose. The purpose of SOA is to lessen artificial dependencies, but this could be removed and maximize real ones. This is via loosely coupling, and the concept of service. A service is a coarse grain functionality objects, with interfaces expressed via a well defined platform like independent language. Upon the usage of services as computational objects, systems can basically register, find and invoke each other based on a well defined, every one accepted, language hence no one, it becomes dependent to another system and a compatible high degree of loosely coupling is achieved.

3. Transforming SOA and MDA to Mobile Business Applications

3.1. Ideas

Ideally, inside a company, the various business and service models will be developed and maintained to to fit on the current status. Combining a service-oriented modeling architecture with MDA can aad benefits to this.

The organized set up of models and information on the stereotypes derived from the service-oriented architecture and select perspective as development process. MS2Web

solution for MDA in our approach is uniquely positioned to take advantage of the unified modeling architecture which results from bringing these two key architectures together.

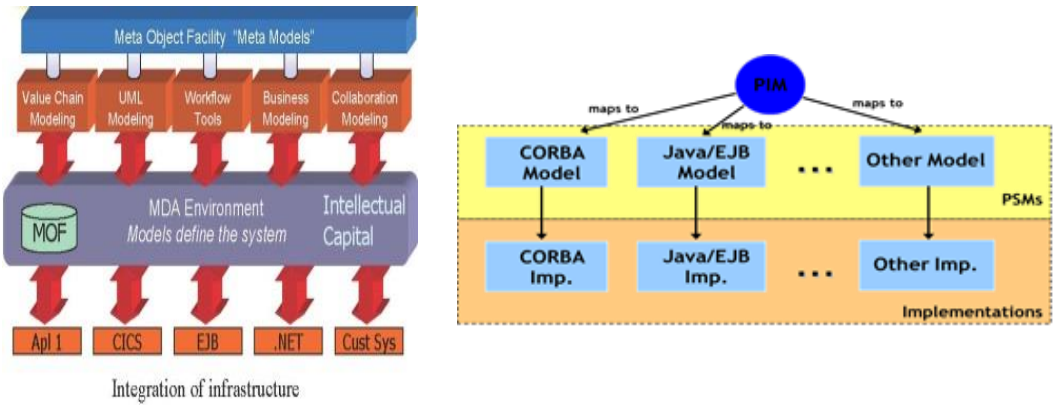

Figure 3. Architecture for Applying MDA to Mobile Business Development Process

Figure 3, As a consequence, it is not realistic to build specific workbenches for the first category of people. By creating the MOF correspond to a subset of UML, they standardized by OMG working groups and some are independently defined by user groups. We give you the architectural model for many elements in Figure 3.

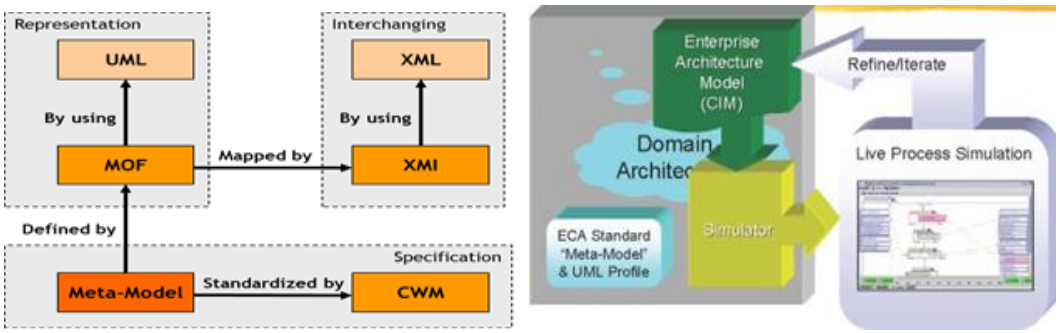

Figure 4. Architecture for Mobile Business Development in this Paper

In Figure 4, PIM of the system is created using UML diagrams by the analyst of the system. The PIM of the system will be designed simply without thinking about services accomplished as CBD (Component-Based Development). The SOA-based PSM would be derived from the present PIM. Creating the PIM, this PIM is transformed -with a transformation tool- to another PIM based on SOA.

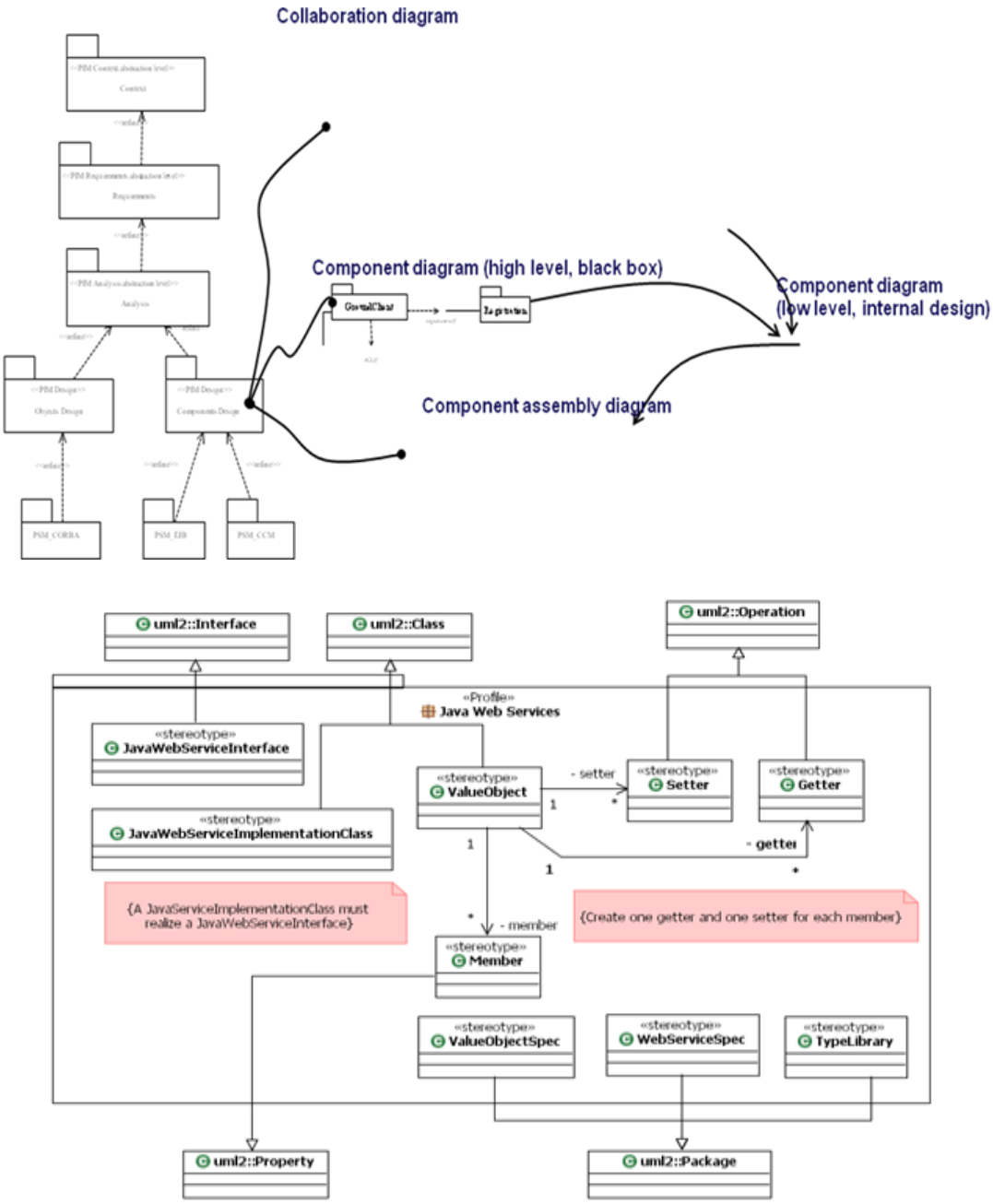

Figure 5. PIM of Mobile Business Applications

3.2. Generating the PIM for Mobile Business

The core of the platform independent model (PIM) is a UML model that is basically from use cases through classes, interactions, states and some UML elements to the components.

3.3. Translation from PIM to PSM

While the PSM entities, *i.e.*, Java classes or entity beans, bear the structure and deliver the behavior of inventory entities in the original inventory PIMs, end-users should not interact directly with these entities, but entities should be accessed through a single interface that exposes a simple set of management methods and hides their complexity. It is a standard design guideline, which conforms to the related design pattern and

influences the architectural design of components. To comply with the guidelines, the case-study aims at implementing an application tool that provides users to manage the inventory content through a simple GUI. Sample users of such a tool may be front-desk operators who respond to customer calls and access the inventory to set-up a new or change the state of an existing product/service instance.

The case-study uses MDA to automatically generate the tool and associated GUI in Java and J2EE (session bean) in order to deliver the required embedded pattern and design guideline. In this study, we only focus on Java outputs.

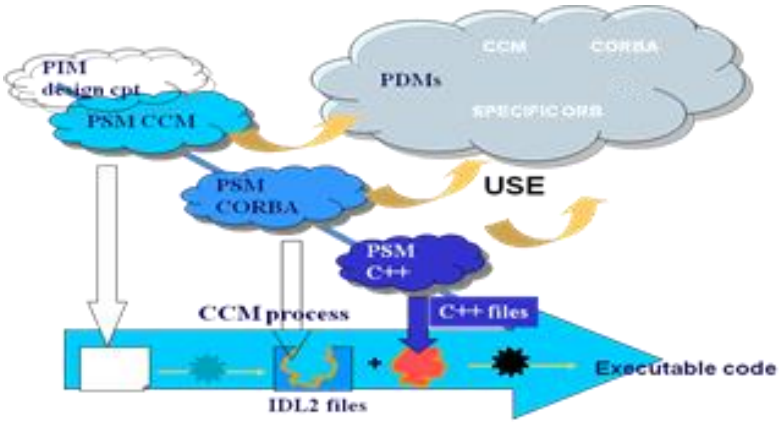

Figure 6. Overall Architecture for PIM to PSM Translation

Figure 6, Explains the overall architecture for PIM to PSM translation. Making PSM based on SOA to the PSM based on mobile business Services using WSDL. This value object and each interface in PIM will be altered to WSDL Type and Port Type in the PSM and the parameters of methods will be transformed to the Messages (Input/Output) in the PSM. A transformation t transforms a model Ma into another model Mb t: Ma -> Mb. Model Ma is supposed based on meta-model MMa and model Mb is supposed based on meta-model MMb. We note this situation as: sem(Ma,MMa) sem(Mb,MMb).

This transformation is like any other model. So model Mt. Mt: Ma->Mb. Obviously since Mt is a model, we postulate the existence of a generic transformation meta-model MMt, which would be similar to any other MOF based MDA meta-model.

```
mapping ParameterToInputPart (in UML2.Parameter) : WSDL.Part {
guard self.direction <> "return" {
        name := self.name;
        element := self.type.resolveByRule(
                "UMLTypeToWSDLElement", WSDL.Element);
        type := self.type.resolveoneByRule(
                "UMLTypeToWSDLType", WSDL.Type);
}

mapping ParameterToOutputPart (in UML2.Parameter) : WSDL.Part {
guard self.direction = "return" {
        name := self.name;
        element := self.type.resolveByRule(
                "UMLTypeToWSDLElement", WSDL.Element);
        type := self.type.resolveoneByRule(
                "UMLTypeToWSDLType", WSDL.Type);
}
```

Figure 7. Example of Translation Interface PIM to PSM

3. Conclusion

As Service Oriented Architecture (SOA) paves it way to the business world. The problems of modeling solutions based on SOA have been set to fix problems Reinforced by the Supply-Manage-Consume, the different modeling of solutions and services is a good practice incorporated into advanced development processes that support SOA, including Select Perspective. Service Interfaces are shared amongst models showing the implementation and re-use of the services.

The vivid organization of models and data based on the stereotypes derived from the service-oriented architecture and Select Perspective as development process. The productivity, quality and impact analysis benefits of the use of MDA with its importance on automation, transformation and synchronization. The select Solution for MDA is uniquely palced to take advantage of the unified modeling architecture which results from bringing these two key architectures together. We introduced an approach to modeling and design of complex distributed systems using SOA and MDA. The PIM of the system is created and then the PSM based on SOA is generated. Then end PSM based on a target platform is generated. MDA has this transformation tool generated though these models.

Acknowledgments

This Research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the C-ITRC (Convergence Information Technology Research Center) support program (IITP-2016-H8601-15-1007) supervised by the IITP (Institute for Information & Communication Technology Promotion).

This Research was supported by the International Research & Development Program of the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (Grant Number: K 2014075112).

References

[1] http://www.softwaretestingclass.com/introduction-to-mobile-application-testing /
[2] N. Tillmann and W. Schulte, Stub-object generation with behavior. In Proceedings of the 21st IEEE/ACM international
[3] Conference on Automated Software Engineering (September 18 - 22, 2006). Automated Software Engineering. IEEE Computer Society, Washington, DC, 365-368., 2006.
[4] A. M. T. G. Basani, A. Bertolino and E. Marchetti, “The Cow_Suite Approach to Planning and DAMTG ving Test Suites in UML Projects”, Proceedings. Fifth International Conference on the Unified Modeling Language - the Language and its applications UML 2002, LNCS 2460, Dresden, Germany, **(2002)**, pp. 383-397
[5] A. M. T. G. Basani, A. Bertolino and E. Marchetti, “The Cow_Suite Approach to Planning and DAMTG ving Test Suites in UML Projects”, Proceedings Fifth International Conference on the Unified Modeling Language - the Language and its applications UML 2002, LNCS 2460, Dresden, Germany, **(2002)**, pp. 383-397
[6] H.323 Standard http://www.microsoft.com/windows/NetMeeting/Corp/reskit/Chapter11/default.asp
[7] T. J. Ostrand and M. J. Balcer, “The Category Partition Method For Specifying and Generating Functional Tests”, Communication of the ACM, vol. 31, no. 6, **(1988)**, pp. 676-686.
[8] M. Paul, CMM v2.0 Draft C, **(1997)**.
[9] Rational Unified Process version 2000.02.10. Rational Software Corporation. On-line at http://www.rational.com/products/rup
[10] UML Documentation version 1.5 Web Site. On-line at
[11] http://www.omg.org/technology/documents/formal/uml.htm

