Modeling of Distributed Systems with SOA & MDA

Haeng-Kon Kim

Abstract— **Along with the boom of Web services and the thriving Model Driven Architecture (MDA), we must consider the growing significance and utility of modeling in the development of software and solutions. The main advantages of MDA are the ability to transform one PIM into several PSMs, one for each 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 touted as the key to business agility, especially when combined with a model-driven approach. Model-Driven Architecture (MDA) is a well-developed concept that fits well with SOA, but until now it has been a specialized technique that is beyond practical application scope of most enterprises.**

In this paper, We describe the initial investigation in the fields of MDA and generative approaches to SOA. Our view is that MDA aims at providing a precise framework for generative software production. Unfortunately many notions are still loosely defined (PIM, PSM, etc.). We propose here an initial exploration of some basic artifacts of the MDA space to SOA. Because all these artifacts may be considered as assets for the organization where the MDA is being deployed with SOA, we are going to talk about MDA and SOA abstract components to apply an e-business applications. We also discuss the key characteristics of the two modeling architectures, focusing on the classification of models that is embodied by each. The flow of modeling activity is discussed in the two architectures together with a discussion of the support for the modeling flows provided by MDA. Our model of framework – a unified modeling architecture – is introduced which illustrates how the two architectures can be brought together into a synergistic whole, each reinforcing the benefits of the other with case study.

Index Terms—**Model-Driven Architecture(MDA),Domain Model, Service-oriented architectures(SOA), Software Process Improvement, Component Based Development, Repository**

I. INTRODUCTION

In order to find an appropriate solution to development and design of those systems an appropriate paradigm seems necessary. The object-oriented and component-based technology has not significantly met the needs of these systems, and may be considered as adding additional complexity to a domain that needs simplification. A new paradigm like service-oriented architecture is necessary. SOA is a paradigm that utilizes services as fundamental elements for developing applications. In order to gain the full benefits of such technology, an effective approach to modeling and designing these complex distributed systems is required. In fact there is not a suitable approach to SOA-based development and little works have been done on this area and most of them are for special applications and specific domains. To exploit the benefits of SOA effectively and duly, we propose an approach that involves MDA into the context[1,2,3,4].

Service-oriented architecture (SOA) is an approach to loosely coupled, protocol independent, standards-based distributed computing where software resources available on the network are considered as Services[3]. SOA is believed to become the future enterprise technology solution that promises the agility and flexibility the business users have been looking for by leveraging the integration process through composition of the services spanning multiple enterprises. The software components in a SOA are services based on standard protocols and services in SOA have minimum amount of interdependencies. Communication infrastructure used within an SOA should be designed to be independent of the underlying protocol layer. Offers coarse-grained business services, as opposed to fine-grained software-oriented function calls and uses service granularity to provide effective composition, encapsulation and management of services.

The problems of modeling solutions based on SOA have largely been resolved through the recognition of the importance of loose coupling and the consequent separation of concerns. Service Interfaces are shared amongst models showing the implementation and re-use of the services. Whilst the use of modeling within SOA is well established, it has suffered from the same issues as modeling in other architectures. The abstraction gap between the level of detail expressed in the model and the level of detail expressed in the code is a key issue. Yet it is the abstraction gap which is one of the key targets for the Model Driven Architecture. It seems likely, then, that if SOA and MDA can work together they will add value synergistically, leading to greater benefits than either architecture provides in isolation. Yet the two architectures are distant in terms of the way they address the issues surrounding modeling. SOA focuses on the stereotypical roles of models based on separation of concerns. MDA focuses on levels of abstraction, defining the role of models within a process. The question of the compatibility of these two model architectures remains open.

The service-oriented architecture (SOA) approach and the corresponding 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

Heang-Kon Kim is with the Department of Computer Engineering, Catholic University of Daegu Kyung San, Daegu, 712-702, Korea (corresponding author to provide phone: 053-850-2743; fax: 053-850-2740; e-mail: hangkon@ cu.ac.kr).

fields of distributed application development (e.g. enterprise application integration, web application development, inter organizational workflow collaboration). The service-oriented paradigm offers the potential to provide a fine grained virtualization of the available resources to significantly increase the versatility.

Model driven architecture (MDA) [7] has been proposed as an approach to deal with complex software systems by splitting the development process into three separate model layers and automatically transforming models from one layer into the other :

1) The Platform Independent Model (PIM) layer holds a high level representation of the entire system without committing to any specific operating system, middleware or programming language. The PIM provides a formal definition of an application's functionality without burdening the user with too much detail.

2) The 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) The Code Layer consists of the actual source code and supporting files which can be compiled into a working piece of software. In this layer, every part of the system is completely specified. MDA theory states that a PIM is specified and automatically transformed into a PSM and then into actual code, thus making system design much easier. The trick, of course, lies in the development of generic transformers capable of generating PSM and code layers from the PIM [8]. e-business application on SOA is a relatively young field of distributed computing and is currently lacking any form of tool support for a model driven approach to software development. This is unfortunate since we believe that due to its high complexity and the high rate of churn in the software technology market, a MDA approach is vital to the adoption of this new technology as in figure 1. 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. Furthermore, the developers responsible for the integration of the middleware into the overall system should be able to concentrate on middleware concerns and not have to cope with the business logic as well. This separation of concerns can be greatly facilitated by an appropriate MDA approach.

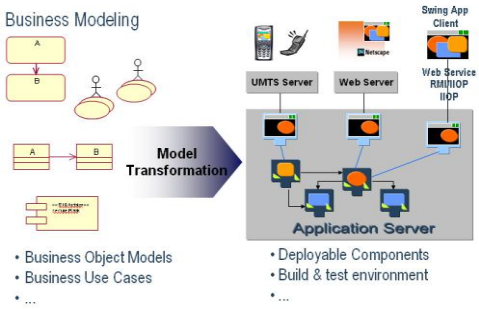

Fig.1. MDA approaches

In this paper, we present 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 is achieved via appropriate tools. The paper proposes a new approach to modeling and designing service-oriented architecture. In this approach the PIM of the system is created and then the PSM based on SOA is generated (this PSM is a PIM for next level). Then the final PSM based on a target platform (such as Web Services, Jini and so on) is generated. These models are generated with transformation tools in MDA and an approach to the model driven development for e-business applications on SOA is presented. The goal of the approach is to minimize the necessary human interaction required to transform a PIM into a PSM and a PSM into code for a SOA. To further separate the architectures specific components of the PSM from the business specific components of the PSM, a UML e-business Profile is introduced and a separation of the PSM layer into two parts is proposed which make the automated transformations from PIM to PSM to code easier to implement and more transparent for system designers, developers, and users. The separation of concerns introduced on the PSM layer is mirrored on the code layer by the use of Java annotations, allowing the same business code to run in different domains simply by exchanging the annotations and thus decoupling application code and SOA middleware.

II. RELATED WORKS

A. Modeling Web services metadata based on MDA

Web services are emerging as the perfect framework for application-to-application integration or collaboration, to make these applications available as Web services. To standardize 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 Web service functionality. Essentially, a WSDL file is a language-independent XML-based version of an IDL (Interface Definition Language) file that describes the operations offered by a Web service, as well as the parameters that these operations accept and return. Thus, WSDL has become 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.

- The **WS-Policy** specification describes 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.

- The **WS-Policy Attachment** specification describes how to associate policies with a particular subject. It

gives normative descriptions of how this applies in the context of WSDL and UDDI, (Universal Description, Discovery, and Integration), and it provides an extensible mechanism for associating policies with arbitrary subjects through the expression of scopes.

Along with the boom of Web services and the thriving Model Driven Architecture (MDA), we must consider the growing 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 that proposes to model the business logic with Platform-Independent Models (PIMs) to later transform them on Platform-Specific Models (PSMs) by using transformation guides between the different models. The main advantages of MDA are the ability to transform one PIM into several PSMs, one for each 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.

Because Web services are software components, the development of Web 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 which the Web service will be deployed, the code that should be generated is the WSDL document that contains the Web service description in a standard format.

B. MDA Main Concepts

The main concepts of the MDA are beginning to be identified [6,7] A model represents a particular aspect of a system under construction, under operation or under maintenance. A model is written in the language of one specific meta-model. A meta-model is an explicit specification of abstraction, based on shared agreement. A meta-model acts as a filter to extract some relevant aspects from 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. The OMG MDA postulates the use of the MOF as the unique metameta-model for all IT-related purposes. The MOF contains all universal features, 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 utilizes models and a generalized idea of architecture

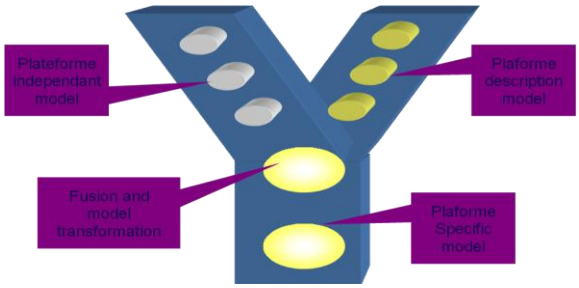

Fig.2. MDA Development Process

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. MDA Meta model acts as a filter to extract 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 show the MDA process that includes: 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 first step in using MDA is to develop a CIM which describes the concepts for a specific domain. For example, a CIM might describe experiment protocols, or properties of genes. The CIM focuses on the environment and 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, network configurations, and programming language. The meanings of PIM and PSM 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 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 describe 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.

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

The MOF contains 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.

C. SOA

SOA 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 musts other systems expose. The target of SOA is to minimize artificial dependencies (although it can never be completely removed), and maximize real ones. This is done via loosely coupling, and the concept of service. A service is a coarse grain functionality objects, with interfaces expressed via a well defined platform independent language. When using services as computational objects, systems can register, find and invoke each other based on a well defined, every one accepted, language hence no one, highly becomes dependent to another system and a high degree of loosely coupling is achieved.

III. APPLYING MDA TO E-BUSINESS APPLICATIONS

A. Basic ideas

The MDA organization may be viewed as a set of artifacts, some being standard building blocks, some being user developed. We may envision, in the not too far future, an organization starting with a hierarchical library of meta-models and extending it as an adaptation to its own local context (models as assets). Model reusability will subsume code reusability, with much more efficiency. This may be seen as orthogonal to code class libraries (e.g. Java, Swing, EJB, etc.). Inside a company, the various business and service models will

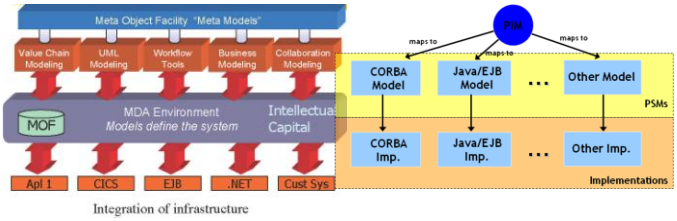

Fig.3. Architecture for Applying MDA to e-business Development Process

be developed and maintained to reflect the current situation. Combining a service-oriented modeling architecture with MDA for e-business can bring many unique benefits. Firstly the clear organization of models and information based on the stereotypes derived from the service-oriented architecture and select perspective as development process. Secondly the productivity, quality and impact analysis benefits of the use of MDA with its emphasis on automation, transformation and synchronization. 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. MDA combines a uniquely powerful implementation of the web services vision, together with the industry leading solutions for modeling service-based solutions.

Figure 3 shows our architecture for applying MDA to e-business and web service application in this paper. First it defines the language used for describing object-oriented software artifacts. Second, its kernel is synchronized with the MOF for practical reasons as previously mentioned. There is much less meta-modelers (people building meta-models) than modelers (people building models). As a consequence it is not realistic to build specific workbenches for the first category of people. By making the MOF correspond to a subset of UML, it is possible with some care to use the same tool for both usages. As a consequence the MDA is not only populated by first class MOF meta-models, but also with UML dialects defined by UML profiles for specific purposes languages. This is mainly done for practicality (widening the market of UML tools vendors) and there is some redundancy between UML profiles and MOF meta models (It is even possible to find conversion tools). There are many examples of profiles. Some are standardized by OMG working groups and other are independently defined by user groups or even by individuals. Examples of profiles are "UML for APL 1", " UML for CICS", "UML for Scheduling Performance and Time" (real-time applications), "UML for EJB", "UML testing", "UML for EAI", "UML for QoS and fault tolerance", "UML for Cust Sys".

One important kind of model that is being considered now is the correspondence model. A correspondence model explicitly defines various correspondences that may hold between several models. In the usual case, there are only two models: the source and the target. There may be several correspondences between a couple of elements from source and target. The correspondences are not always between couples of elements and they are strongly typed. There is not yet a global consistent view on correspondence models since this problem is appearing

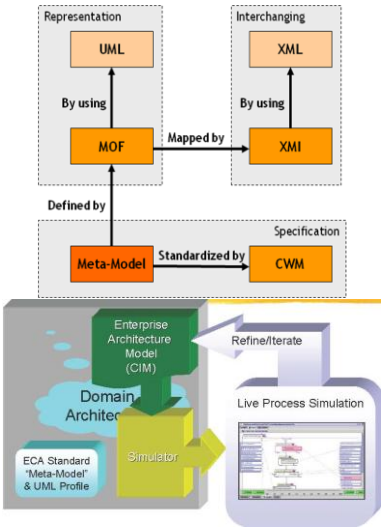

Fig.4. Architecture for e-business Development in this paper

from different perspectives. When the notion of PDM and virtual machine is clarified we may then tackle the definition of a PIM, a model containing no elements associated to a given platform. In other times this was simply called a business model, but as for platform models we need to progress now towards a less naive and a more explicit view. The first idea is that the PIM is not equivalent to a model of the problem. We propose the architectural model for many elements of the solution that may be incorporated in a PIM as long as they don't refer to a specific deployment platform as in figure 4.

In our architecture model as in figure 3, the 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 that is pretty simple and is accomplished as CBD(Component-Based Development). The SOA-based PSM (which is a PIM for the next level) would be derived from the present PIM. The way which is used to identify this PSM must be quite different from the one used to identify PSM in component-based systems; because in componentbased systems the patterns which are used to determine the PSM of the system have a specific form. For each service in e-busimess applications, there is a single instance which manages a set of resources and consequently, unlike components, services are for the most part stateless that means need to view a service as a manager object that can create and manage instances of a type, or set of types. According to above discussion, in our approach after creating the PIM, this PIM is transformed -with a transformation tool- to another PIM based on SOA. In this transformation, for each class diagram in PIM for e-business, a Service Manager is created that manages the Instant Services. This management involves creation, deletion, updating a service and state management of services. To complete this transformation, we need some other special patterns for dealing with associations between classes. When this PIM based on SOA is created, the PSM of the system can be created based on a target platform such as Web Services, e-business and/or other platforms with transforming tools. Some operations apply on a

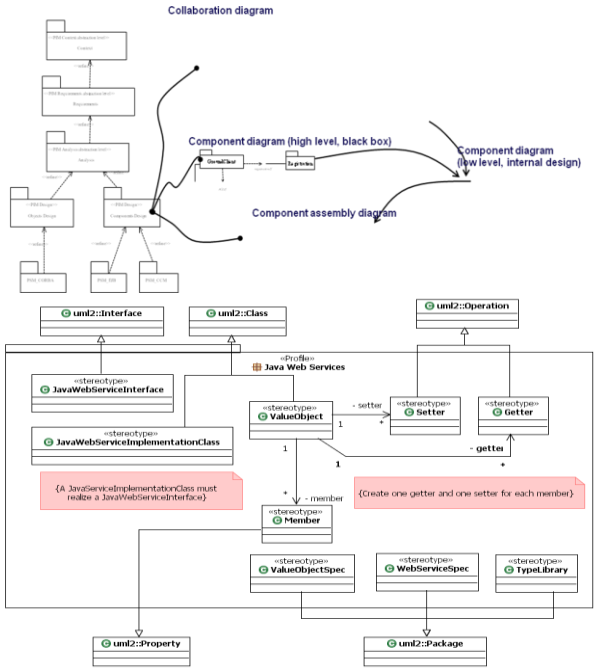

Fig.5. PIM of e-busincess applications

single model and are called monadic by opposition to dyadic operations applying to two models. Operations applying on more than two models are more rare. Obviously the most apparent components in an MDA workbench are the precise tools composing this workbench. Fortunately in this context we should be able to propose a rather precise definition of a tool: it is an operational implementation of a set of operations applicable on specific models. The meta-models supported by a tool should be exhaustively and explicitly defined.

B. Generating the PIM for e-business

PIM for e-business application is an abstract design of a computerized solution which does not include any platform specific elements. The core of the platform independent model (PIM) is a UML model – ranging from use cases through classes, interactions, states and other UML elements to the components as in figure 5.

C. Translation from PIM to PSM

While the PSM entities, i.e. Java classes or entity beans, bear the structure and deliver the behaviour of inventory entities as described in the original inventory PIMs, end-users should not interact directly with these entities. Rather, entities should be accessed through a single interface that exposes a simple set of management methods and hides their complexity. This is a standard design guideline, which conforms to the related design pattern and influences the architectural design of components. In order to comply with the guideline, the case-study aims at implementing an application tool that allows users to manage the inventory content through a simple GUI. Example users of such a tool may be front-desk operators who respond to customer calls and access the inventory to setup a new or change the state of an existing product/service instance. The case-study

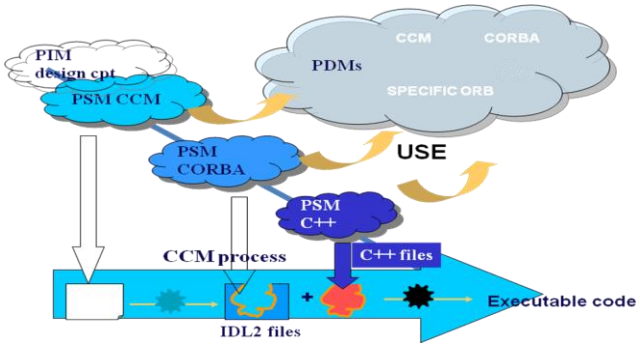

Fig. 6. Overall architecture for PIM to PSM translation

```
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);
}
```

Fig.7. Example of translation Interface PIM to PSM

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. Again, this paper only concentrates on the Java outputs.

Figure 6 shows the overall architecture for PIM to PSM translation. Transforming PSM based on SOA to the PSM based on e-business Services using WSDL is a straightforward task. In our approach, each value object and each interface in PIM will be transformed to WSDL Type and Port Type in the PSM respectively 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) As a matter of fact, a transformation is like any other model. So we'll talk about the transformation 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:

In some cases the transformation takes some particular form if the source and target meta-models are in the relation of refinement like a CORBA and a CCM meta-model. Figure 7 show the examples of translation interface.

IV. Conclusion

Service Oriented Architecture (SOA) is increasingly important in the business world as b2b transactions become ever more vital to business process out-sourcing and other co-operative activity. The problems of modeling solutions based on SOA have largely been resolved through the recognition of the importance of loose coupling and the consequent separation of concerns. Reinforced by the Supply-Manage-Consume concept, the separate modeling of solutions and services is a well established 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.

Whilst the use of modeling within SOA is well established, it has suffered from the same issues as modeling in other architectures. The abstraction gap between the level of detail expressed in the model and the level of detail expressed in the code is a key issue.

Yet it is the abstraction gap which is one of the key targets for the Model Driven Architecture.

Combining a service-oriented modeling architecture with MDA can bring many unique benefits. Firstly the clear organization of models and information based on the stereotypes derived from the service-oriented architecture and Select Perspective as development process. Secondly the productivity, quality and impact analysis benefits of the use of MDA with its emphasis on automation, transformation and synchronization. Select Solution for MDA is uniquely positioned to take advantage of the unified modeling architecture which results from bringing these two key architectures together.

In this paper we introduced an approach to modeling and design of complex distributed systems using SOA and MDA. In fact, to exploit the benefits of SOA effectively and duly, we propose an approach that involves MDA into the context. In this approach the PIM of the system is created and then the PSM based on SOA is generated. Then the final PSM based on a target platform is generated. These models are generated with transformation tools in MDA.

References

[1] Jean Bezivin, "Slimane Hammoudi, Denivaldo Lopes and Frederic Jouault. Applying MDA Approach for Web service Platform," Proceedings of the 8th IEEE Intl Enterprise Distributed Object Computing Conference, 2004.

[2] Michael N.Huhns, Munindar P.Singh, "Service-Oriented Computing: Key Concepts and Principles," Journal of IEEE Internet Computing, 2005.

[3] Adel Torkaman Rahmani, Vahid Rafe, Saeed Sedighian, and Amin Abbaspou, "An MDA-Based Modeling and Design of Service Oriented Architecture," ICCS 2006, Part III, LNCS 3993, 2006, pp. 578 – 585.

[4] Aniruddha Gokhale and Balachandran Natarajan(2002). Composing and Deploying Grid Middleware Web Services Using Model Driven Architecture. *In Lecture Notes in Computer Science*. Volume 2519, pp. 633–649. Available: http://www.cydex21.com

[5] Rakesh Radhakrishnan and Mike Wookey, "Model Driven Architecture Enabling Service Oriented Architectures," In Whitepaper SUN Microsystems, pp.1 – 13, 2004.

[6] David Skogan, Roy Gronmo, and Ida Solheim, "Web Service Composition in UML," Proceedings of the 8th IEEE Intl Enterprise Distributed Object Computing Conference, pp. 111, 2004.

[7] Qusay H. Mahmoud, "Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application integration (EAI)," Sun Developers Network, Sun Developers network, April 2005

Dr. Haeng-Kon Kim is currently a professor in the Department of Computer Engineering and was a dean of Engineering College at Catholic University of Daegu in Korea. He received his M.S and Ph.D degree in Computer Engineering from Chung Ang University in 1987 and 1991, respectively. He has been a research staff in Bell Lab. in 1988 and NASA center 1978-1979 in U.S.A. He also has been reserched at Central Michigan University in in 2000-2002 and 2007-2008 in U.S.A. He is a member of IEEE, KISS and KIPS. Dr. Kim is the Editorial board of the international Journal of Computer and Information published quarterly by ACIS. His research interests are Component Based Development, Component Architecture & Frameworks for Mobile Applications and Components embedded systems .