Implementation of MDA Method into SOA Environment for Enterprise Integration

Wiranto Herry Utomo

Faculty of Information Technology, Satya Wacana Christian University
Salatiga, Central Java, Indonesia

Abstract

Even though SOA provides real contribution, it is not adequate to implement enterprise integration. There are still problems in the implementation of enterprise integration in SOA environment, they are 1) the absence of modeling language support, 2) the absence of guideline of the services implementation produced by services identification, and 3) service orchestration that uses only Web Services. Based on the consideration and comparison of some integration methods, MDA method from OMG is chosen as a method to help dealing with SOA weaknesses. Model-driven based MDA method enables business level functionality to be modeled by UML language modeling that is separated from low level implementation (code level). Therefore, SOA used in MDA approach can be expressed using UML modeling language. This study proves that SOA-MDA method has been successfully used to perform analysis, design and implement of enterprise integration.

Keywords: *SOA, MDA, UML, Web Services, Integration.*

1. Introduction

SOA is a framework in company architecture and aims at achieving the same business' goals: minimize ownership costs and create flexible business solutions that improve business' stability, reduce time to the market and provide support for global expansion. SOA substantially impacts the whole key aspects of enterprise architecture. Business service proposed by SOA forms the basic of business architecture and process architecture.

SOA forms business architecture because business' functions are exposed as services that can be divided and reused. Business process, services and event are converted to appropriate application services that create and support services architecture. Services alone form application architecture, whereas information architecture is achieved through data standardization and data availability through interface services [17]

SOA is a software architecture designed based on service oriented design principles [3][15][5][8][13][7], whereas service orientation is a concept in software engineering that represents different approaches to separate interest.

According to Erl [3], in general, software that does not use SOA can be divided into two main layers, Application Layer where application runs and Business Process Layer that describes how business process in a company runs. Organization business process will be defined in application along with technical program code. In SOA implementation, service oriented process is implemented in a layer between Business Process and Application Layer where both are parts of logic enterprise. The layer is called Service Interface Layer and can be seen in Figure 1.

This layer is to wrap the logic in Application Logic along with the business process in Business Logic. Through this approach, application can be more modularized with more varied technology.

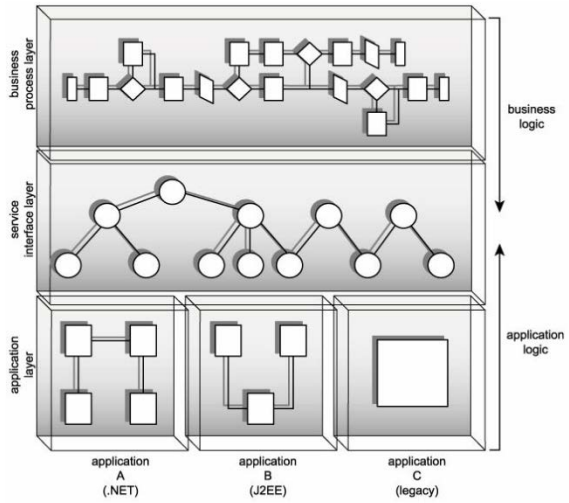

Fig 1. Service Interface Layer in SOA method [3]

Services analyses and identification can be done through this SOA method [3]. The services achieved then mapped to Service Interface Layer, which are Application Layer, Business Process Layer and Service Interface Layer (Figure 1).

Even though SOA [3] provides real contribution, it is not adequate to implement enterprise integration. There are still problems in the implementation of enterprise integration in SOA environment, they are 1) the absence of modeling language support, 2) the absence of guideline of the services implementation produced by services identification, and 3) service orchestration that uses only Web Services.

Orchestration using Web Services has two weaknesses, in terms of scalability and the inability to deal with protocol and data discrepancy. To deal with this, Web Services orchestration with ESB has now been developed. ESB is an infrastructure for SOA service connection and message exchange. ESB main functionality is to do routing, protocol transformation and message or data transformation. Protocol and data discrepancy can be overcome by the protocol and data transformation in ESB. ESB eases connection and mediation, simplifies integration and eases the reuse of service components that lead to a high scalability integration.

Other strengths of Web Services orchestration with ESB is that it enables business layer and information system to have a closer relation because Web Services orchestration is presented in high abstraction level called business process by hiding middleware traditional object used to support business to business interaction. Aside from that, business requirements can be directly translated into business process application through Web Services composition. That is why SOA method alone is not yet optimal to implement enterprise integration. Thus, other methods that are able to deal with the method's weaknesses are required.

Based on the consideration and comparison of some integration methods, MDA method from OMG is chosen as a method to help dealing with SOA weaknesses. The decision to choose MDA method to be combined with SOA method is based on: 1) MDA method is a model-driven method based on the use of platform independent technology model, 2) this method can be used to transform high level business process model to low level one (code), 3) the existence of standard modeling language, 4) this method has used ESB as midleware infrastructure, 5) the phases of the process in this method use system development life cycle.

Model-driven based MDA method [9][4] enables business level functionality to be modeled by UML language modeling that is separated from low level implementation (code level). Therefore, SOA used in MDA approach can be expressed using UML modeling language.

By combining MDA and SOA methods, two completing each other advantages will be gained. SOA provides an infrastructure that reduces complexity in the services reuse and integrates all kinds of technology, protocol, and application whereas MDA is used in High Level Business Process Model transformation to platform independent low level one (programming code). The integration of SOA and MDA methods will be a complete method for enterprise integration..

2. Related Works

Rafe et al [12] stated that the goal of their research are to provide a successful and usable conjunction between these two technologies. They have tried to provide a simple yet effective process which can be viewed as a framework. In the vision inspired by this framework, SOA is the product and MDA makes its production line. During this process, input model is provided via XMI standard and with a high level of abstraction. Proposed framework analyses the elements and their relations within the given model and tries to recognize the SOA components. In two phases (Figure 2), the input model is first transformed into a SOA profile based model and then into a middleware independent code. Middleware transparency is achieved via the concept of Aspect. The final phase of framework is to transform middleware transparent code into an executable code based on one of known middlewares for SOA. Jini middleware and pre-process weaving is used in the last phase.

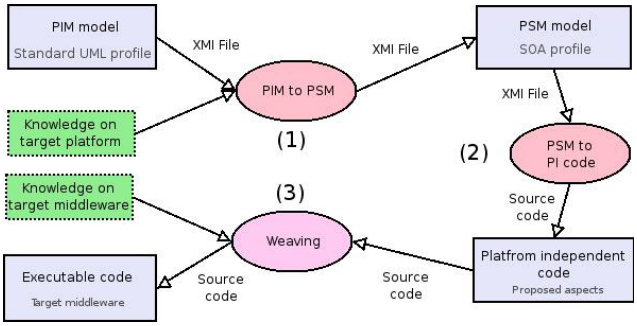

Fig 2. Framework Components [12]

The phase of PIM to PSM can be considered as the most important and complex part of the framework. In this step, the platform independent model - based on UML standard profile - is transformed to the platform specific model - based on proposed SOA profile. Although we have tried to apply MDA to SOA for simpler model, the approach taken here has more capabilities and can handle relatively more complex cases.

In this approach, the input model (PIM) has no direct information about SOA. Obviously using such an abstract input - based on standard UML - requirements a more autonomous model transformer. By autonomous we mean

a model transformer which tried to depend on the specification of model rather than human guidelines. Such a model transformation is beyond what we expect from an MDA based model transformer and also beyond most of the current frameworks.

Model-Driven Architecture (MDA) is proposed by the Object Management Group (OMG) as a reference to achieve wide integration of enterprise models and software applications. MDA is a best choice to address how SOA should be designed, developed and integrated. MDA provides specifications for an open architecture appropriate for the integration of systems at different levels of abstraction and through the entire information systems' life-cycle. The MDA comprises three main layers: Computation-Independent Model (CIM), Platform-Independent Model (PIM), Platform Specific Model (PSM) . MDA lies in separating the enterprise model from the technology infrastructure, making a clear division between the business functions and the implementation details.

The Computation Independent Model (CIM) cares about the requirements for the systems by describing the situation in which the system will be used. Such a model is sometimes called a domain model or a business model and hides information about the use of automated data processing systems.

The Platform-Independent Model (PIM) describes the operation of a system while hiding the details necessary for a particular platform. The model focus on specifications that are not changing from one platform to another e.g. BPMN (independent from Workflow engine) or UML (independent of computing platform).

A Platform-Specific Model (PSM) combines the specifications in the PIM with the details that specify how theses systems are using a specific type of platform.

There are three levels, CIM, PIM and PSM, in MDA method according to OMG. In the research conducted by Rafe et al [12], CIM level is not used but is directly jumped to PIM-PSM level. Aside from that, in Rafe et al research [12], SOA is used after modeling of the PIM-PSM level. Therefore, therefore two differences of this research and Rafe et al research [12] : 1) this research used three complete MDA levels, CIM, PIM and PSM, 2) SOA is combined in CIM level to MDA in order to decide business process and identify services, MDA modeling in the next level, PIM and PSM, is done after the services found.

3. SOA – MDA Methods

The integration OF SOA and MDA complete each other and cover each other weaknesses. Actually, the use of 'integration' term is not appropriate. The more appropriate term is MDA 'implementation' into SOA environment. The application of model driven method in SOA environment is phases of high level Business Process model into executable services and can be orchestrated into the integration of services. Phases or processes of the integration method refer to Object Oriented System Development Life Cycle that refers to Solamo [14]. Phases of SOA-MDA method can be seen in Figure 3.

New method as a result of integration proposes service oriented approach for integration by determining two factors:

- Business Perspective focuses on business features and requirements from the application that will be constructed.
- System Perspective focuses on functionality and process requirements to be implemented in the application to satisfy business requirements.

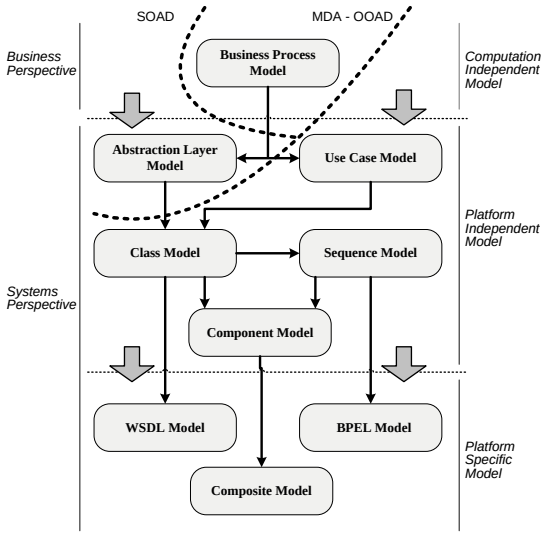

Fig. 3 New method of the integration of SOA and MDA method

This integration method provides a series of concepts required for modeling of the two perspectives. All concepts can be seen in Figure 3 that represents the two methods, SOA and MDA. The concepts related to business perspective explain the attached elements in business and are represented in CIM Model through Business Process Model. The concepts related to system perspective are elements used to describe system functionality and process and are represented in PIM dan PSM Model, with Use Case Model, Class Model, Sequence Model, Component Model, WSDL Model, BPEL Model and Composite Model.

Some barriers in the selection of programming language, hardware, network typology, communication protocol, infrastructure, etc occur in software development. Each element is considered as a part of the platform solution. CIM approach helps to focus on essential part of the solution designed, separated from platform details. CIM does not show structure details of the system. CIM plays important roles in bridging the gap between domain expert and its requirements, as well as the experts who constructs artifacts who work side by side to fulfill the domain requirements. CIM is referred as business domain model that explains the knowledge of business domain, which is free from business process or particular software used [1][16][6][10].

In CIM level, which is business analysts oriented, this method uses Business Process Model by adopting BPMN notation that is in fact the standard of business process modeling. However, the business process modeling also uses Activity diagram beside BPMN notation. Business Process Model is used to define identification guideline and business concept representation. This Business Process Model eases service identification to be implemented into application, and transform it into low level one such as PSM level to model Web Services and its composition.

PIM is a view of a system from platform independent point of view. PIM indicates particular levels of platform-freedom so that it can be used for some different platforms. PIM can be seen as the specification of free technology system functionality that will be used to implement the functionality. PIM provides formal specification from system structure and function that is free from any platform. From this point of view, it can be said that CIM is a component of PIM since platform independent component describes computational component and its independent interaction. This components and interface are ways to realize some more abstract information system or application, which automatically help to create a CIM [1][16][6][10].

PSM explains how particular technology can be used to implement the function described in PIM. PSM is adapted with the system in term of implementation construction provided by a particular implementation technology. PSM possesses components for target platform. PIM can be transformed into one or more PSM. Particular platform is produced for every particular technology platform [1][16][6][10].

According to OMG [9][4], PSM is a system reviewed from a particular platform point of view. For Example, Class Model is PIM with service implementation architecture choice, if the model chooses to use particular service technology such as Web Services, the Class Model is then transformed into specific PSM for Web Services.

The use of UML can be said as a common thing in most methods. However, this integration method proposes the use of UML for modeling notation in PIM level, whereas modeling in PSM level will be adapted with implementation platform.

This integration method is a complete development method because it views modeling from all elements related to services oriented system development. This method does not only include one level in driven model, such as PIM, or PSM, but it includes all level including CIM, PIM dan PSM.

4. PHASES OF SOA-MDA METHOD

Figure 3 shows that the processes start by building Business Process Model that later result in services. These processes include several phases where each phase is related to the “creation” of different models. The phases of this integration model include nine phases 1) Business Process Model, 2) Abstraction Layer Model, 3) Use Case Model, 4) Class Model, 5) Sequence Model, 6) Component Model, 7) WSDL Model, 8) BPEL Model and 9) Composite Model. The phases of development process of this integration method are:

1. Business Process Model

With the fast changing business, company can build new business processes by running existing application. This model includes in business perspective and CIM layer. This model is a high level model that serves as the modeling starting point of this method. This model is derived from SOA and MDA methods. The notation used in Business Process Model can use either BPMN or Activity Diagram UML notation.

Business Process Model does not only focus in individual business process representation, which can be fulfilled by workflow description using BPMN that are implemented into WS-BPEL, but also focuses in the development of SOA solution using business process. Business Process Model manages services in workflow business context. This model displays service management in top-down process level. Top-down direction eases the mapping of business requirements into tasks that include activity flows, every is activity realized by existing business process and service components.

To decompose business process, first, tasks are broken down into smaller ones and then map each business process into services.

By implementing SOA design and method, company business processes are modeled and processes blocks that can be grouped into services are identified. Legacy application is analyzed based on its functionality and is mapped to the services. New service is constructed when there is business process that cannot be mapped to legacy application.

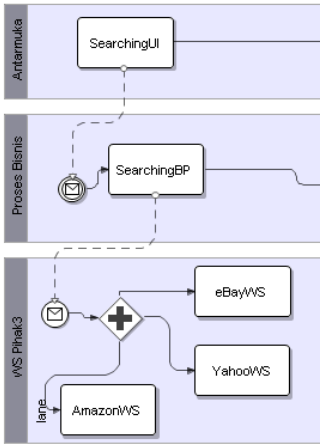

Fig. 4 Process Model using BPMN notation

2. Abstraction Layer Model

Services decomposition can be done from Business Process Model by decomposing business process into the smaller ones. Process Model can be generated into services required to construct new business process. These services can be developed from legacy application, third party or constructing new services.

Abstraction Layer improves Web Services group concept that is a group of Web Services that serves business function as general. Web Services can be published by different service provider and be differentiated from others through specific features. Service layer shows top-down or bottom-up service layer handling.

Even though this integration method is the integration of SOA methods from Erl [3], there are some differences in this Service Layer Model. These differences occur because this integration method uses middleware ESB via BPEL to do Web Services orchestration, whereas in SOA method, Web Services orchestration still uses Web Services (in Service Layer Orchestration).

Service candidate identification is carried out in every layer in SOA, which exists in Application Service Layer, Business Service Layer, and Orchestration Service Layer. However, as mentioned before, ESB replaces Orchestration Service Layer. Therefore, services candidate identification is only carried out in two SOA layers, Application Service Layer and Business Service Layer. Service candidate identification is carried out based on the requirements derived from application Use Case Model and Business Process Model.

In Business Service Layer, identification is carried out by looking at the existing business process and the parts nominated as service candidates are identified. The identification process of service candidate in this layer can be done through task-centric business service. In identifying task-centric business service, identification from Use Case Model in Use Case Diagram is also done in addition to the use of existing business process. Task-centric business layer is gained by mapping the existing phases in business process into services.

Based on Business Process Model constructed, two kinds of services in this layer can be seen 1) input services that create trigger toward business process, and 2) output services that promote invoke. Rademakers-Dirksen [11] explicitly divides these services into two, inbound service to carry out input connection configuration and outbound to carry out output connection configuration. Referring to Rademakers-Dirksen [11], there are two kinds of services in this layer, inbound service and outbound service.

Therefore, in this integration method, Abstraction Layer Model is the improvement of Erl layer model (2005), with the following improvement:

1. Orchestration Service Layer is improved into Service Bus Layer
2. Leave out Application Service Layer. This layer is left out because by the implementation of ESB for integration, all services are services related to business process, and there is no services that technically related to only Application Layer.
3. Business Service Layer is grouped into two service layers, Inbound Service Layer and Outbound Service Layer.

By the use of this new Abstraction Layer Model, the former services found are mapped into Abstraction Layer Model as shown in Figure 5.

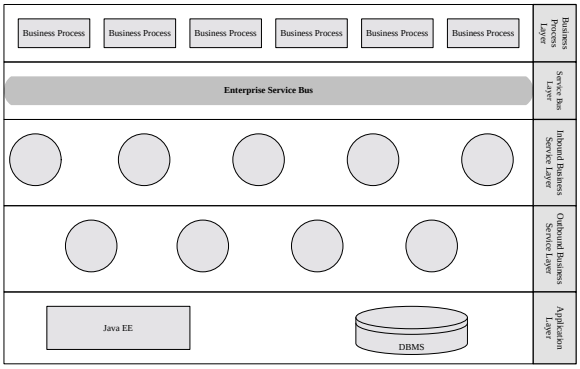

Fig. 5 Abstraction Layer Model

3. Use Case Model

This Use Case Model describes the system's requirements. This phase is a form of software engineering that enables developer in understanding problem domain. This Requirement Model is a series of tasks to know the impact of software development, what the costumers want, and how end users will interact with software.

Use Case Model is used to describe what the system will do, system's functional requirements, and the expected system functionality along with its environment. Complement specification is the not-yet-mapped requirement into Use Case specification that includes non functional requirements such as code maintenance, performance reliability, and system supports or obstacles as well as safety. Use Case Model is a mechanism to achieve expected system behavior without determining how behavior system is implemented.

The output produced in this phase is in the form of Use Case Model (Use Case Diagram), and Use Case Specification. This Use Case model is usually gained based on user's requirements, however, in this integration method, Use Case is produced from Business Process Model. Every business process, which is a functionality of a business unit, is represented into every Use Case of Use Case Diagram.

Use Case Diagram consists of Actor and Use Cases. This diagram shows system functionality and actor communicate with the system. Every Use Case in the model explains the details of the use of Use Case specification. Use Case UML diagram is used as modeling tool for Use Case model. This Use Case diagram consists of three components (see Figure 7):

1. Actor, represents a series of role played by users or system when they interact with Use Case. The actor calls the system to send a service. This Actor can be human or other system. Actor is named after nouns.
2. Use Case, describes the function displayed by the system when it interacts with the Actor. This Use Case is described using verbs or verb phrases.
3. Association, shows the relation or association between Actor and Use Case and or inter-Use Case

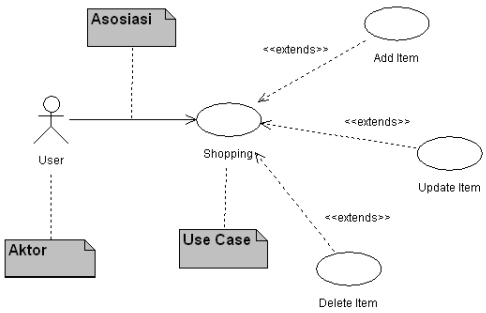

Fig. 6 The Example of Use Case Diagram

4. Class Model

Class model is constructed using UML Class Diagram. This Class Diagram is an input for the following program development. This Class Model represents previous conceptual model over something in the system that possesses behavior. Class Model is an important model in software development because it will be the main input for the following phase. This model is described in Class Diagram containing classes that provide former conceptual model for things in the system that possesses property and behavior. This Class Diagram consists of Boundary Class, Control Class, Entity Class and Web Services Class.

There are four perspectives used in identifying classes, they are boundary between system and actor, the information used by system, and control logic of the system. These four perspectives are described into classes, they are: Boundary Class, Control Class, Entity Class and Web Services Class (Figure 7).

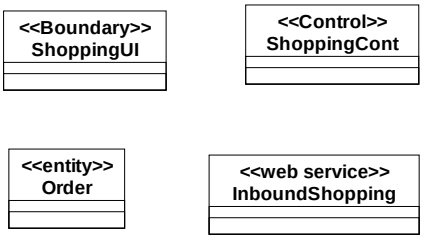

Fig. 7 The Example of Boundary Class, Control Class, Entity Class and Web Services Class

Boundary Class is used to model the interaction between environment and the system working in it. This Class explains system boundary and starting point in identifying related services. This Class limits external power from internal mechanism and vice versa. This Class bridges interface and something outside the system that consists of interface users, system interface and interface tools.

Boundary Class is derived from Use Case Diagram, originally from the set of Actor and Use Case.

Control Class represents system functionality. This Class provides behavior that defines control logic and transaction in Use Case, contributes small change if Entity Class' structure or behavior changes, uses or governs some Entity class' contents. This Class provides system coordinated behavior and limits Boundary Class and Entity Class.

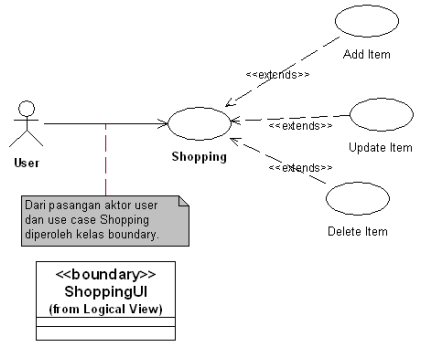

Fig. 8 Boundary Class derived from the set of actor and Use Case

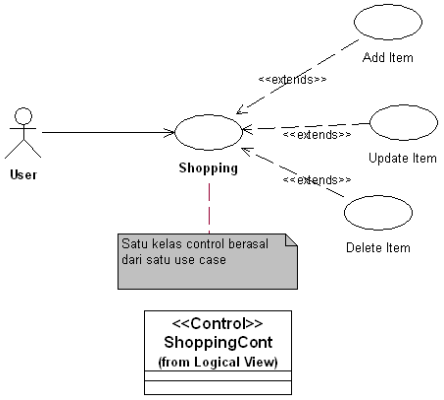

Fig. 9. Control Class from Use Case

Entity Class represents information storage in the system. This Class is used to update information, such as events, phenomena or objects. This class responsible for storing and managing information in the system that represents key concept from the system constructed. This Class Entity can be derived from key abstraction of Use Case Diagram by filtering noun.

Web Services Class is a representation of services found in Service Layer Model.

5. Sequence Model

Sequence Model is created using UML Interaction Diagram containing Collaboration Diagram and Sequence Diagram. Interaction diagram models dynamic characteristics of objects in groups of classes. This diagram models system behavior as how system responds toward a particular user's action, how an object is created or changed as well as how data is transformed.

This model shows interaction and collaboration among analyses classes. Two basic elements used in Behavior Model are object and message. Object is an instantiation of a class, whereas message is a form of communication among objects. This service Process Model can be seen as a collaboration among the object of classes in Service Model. Therefore, the input from this Sequence Model is derived from Class Model. An example of Sequence Model can be seen in Figure 10.

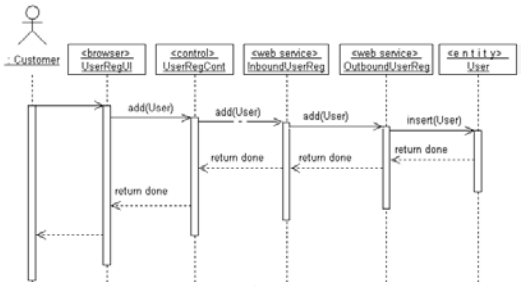

Fig. 10 An Example of Service Process Model

6. Component Model

This model expands the representation of Class Model and the Sequence Model modeled before. Component Model represents components from Web Services composition that identifies services collaborating with business process. This method represents this model using Component Diagram.

This Component Diagram describes components integration which in the next phase will be derived into composite application. The example of Component Model can be seen in Figure 11.

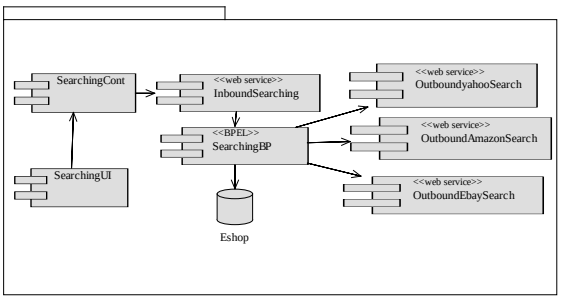

Fig. 11 An Example of Component Model

7. WSDL Model

This model is used to describe Web Services interface that will be used to deliver every services provided by the system. This model is based on WSDL standard. WSDL is a language proposed by W3C to describe Web Services and enables it to describe the interface of services in XML

format. WSDL Model allows to derive graphic representation of Web Services interface that will be generated into WSDL code automatically. The example of Web Service Interface Model implemented in Java EE platform using Netbeans.

8. BPEL Model

This model expands service composition identified by process model explained above by adding particular Web Services based platform details. This Process Execution Model is represented in the form of WS-BPEL. WS-BPEL is a Web Services extension used to facilitate modeling process and BPEL execution in Web Services. BPEL is a modeling language in XML format used to describe business process. The model produced by this language is later executed by BPEL engine.

The explanation of elements in this language will be explained as follow [3]:

1. Process. Process is BPEL's main element. The name of process is defined as name attribute. Aside from that, this tag is also used to insert information related to process definition.
2. PartnerLink and partnerLinks. This element defines the kinds of port from other services involved in business process execution.
3. Variables. This element is used to keep status information used during the process of workflow logic.
4. Sequence. This element organizes a group of activities that they can be executed in an orderly manner. Whereas the elements are supported by WS-BPEL for sequence such as recieve, assign, invoke, and reply.

Beside the four main elements above, WS-BPEL also facilitates some other tags. Standard from BPEL is defined by OASIS and can be achieved from OASIS website. The example of BPEL Model implemented in BPEL using Netbeans can be seen in Figure 12.

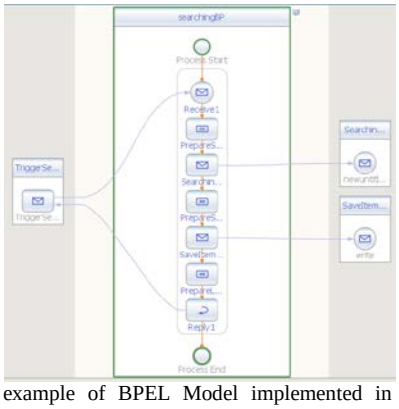

Fig. 12 An example of BPEL Model implemented in BPEL using Netbeans.

9. Composite Model

Composite application (SOA composite application) according to Binildas [2] is an SOA application containing some components such as services, BPEL process, ESB mediation, rules, adapter, and etc. All components must cooperate and support one or more Composite Application.

This model is SOA application modeling containing some components such as BPEL process and ESB. All the components cooperate and support one or more Composite Application. Composite application is the integration of services containing business function and information from a separate source of information. Composite application is a form of integration and application development. Specifically, Composite Application is constructed to support company business process and map it to underlying information resources. In business integration, Composite Application is the final product of SOA. The example of Composite Application can be seen in Figure 13.

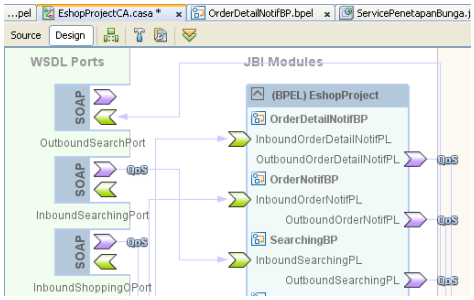

Fig. 13 An example of Composite Application implemented in CASA using Netbeans.

4. Conclusions

The case study to proves thisn methods consists of e Shop application where consumers can shop and place orders for goods offered for sale there. The e-Shop doesn't store inventory but it relies on third parties to warehouse and ship the goods. The third party consisted of the Amazon, Ebay and Paypal. As soon as the e-shop receives an order, it creates a purchase order and sends it to the backend purchasing system which, in turn, sends orders out to one or more suppliers for fulfillment.

This case study of e Shop application proves that SOA-MDA method has been successfully used to perform analysis, design and implement of enterprise integration.

References

[1] Almeida, J.P.A., 2006, Model-Driven Design of Distributed Applications, Ph.D. Thesis, Centre for Telematics and Information Technology, University of Twente, Netherlands

[2] Binildas, C. A., 2008. Service Oriented Java Business Integration, Birmingham-Mumbai: Packt Publishing.
[3] Erl, T., 2005, Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall PTR, Upper Saddle River, New Jersey 07458
[4] Frankel, D.S., 2003, Model Driven Architecture : Applying MDA to Enterprise Computing, Wiley Publishing, Inc., Indianapolis, Indiana
[5] Kanchanavipu, K., 2008, An Integrated Model for SOA Governance An Enterprise Perspective, Master Thesis, IT University of Göteborg Chalmers University of Technology and University of Gothenburg, Göteborg, Sweden
[6] Kim, H., 2008, Modeling of Distributed Systems with SOA & MDA, IAENG International Journal of Cumputer Science, 35:4, 20 November 2008
[7] Li, G., Muthusamy,V. and Jacobsen, H., 2010, A Distributed Service-Oriented Architecture for Business Process Execution, ACM Transactions on The Web, Vol. 4, No. 1, Article 2, Publication date: January 2010.
[8] Nikayin, F.A., 2009, Adopting A Theoretical Method For The Development Of A Service-Oriented Information System, Dissertation, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur
[9] Pastor, O. and Molina, J.C., 2007, Model-Driven Architecture in Practice A Software Production Environment Based on Conceptual Modeling, Springer-Verlag Berlin Heidelberg
[10] Pokraev, S.V., 2009, Model-Driven Semantic Integration of Service-Oriented Applications, Ph.D. Thesis, Centre for Telematics and Information Technology, University of Twente, Netherlands
[11] Rademakers, T., dan Dirksen, J., 2009, Open Source ESBs in Action, Manning Publications Co., Greenwich, CT 06830
[12] Rafe, V., Rafeh, R., Fakhri, P., and Zangaraki, S., 2009, Using MDA for Developing SOA-Based Applications, International Conference on Computr Technology and Development, IEEE Computer Society
[13] Reddy, V.K., Dubey, A., Lakshmanan, S., Sukumaran, S. and Sisodia, R., 2009, Evaluating legacy assets in the context of migration to SOA, Software Qual Journal (2009) 17:51–63, Springer Science+Business Media
[14] Solamo, R., Antonio, J., Asrani, N., Chen, D., de Guzman, O., Feria, R., Petines, J.P., Shin, S., Srinivas, R., Thompson, M. and Villafuerte, D., 2006, Software Engineering, Java Education & Development Initiative, Sun Microsystem.
[15] Sterff, A., 2006, Analysis of Service-Oriented Architectures from a business and an IT perspective, Master Thesis, Technische Universität München, Fakultät für Informatik
[16] Vidales, M.A.S., García1, A.M.F., and Aguilar, L.J., 2008, A new MDA approach based on BPM and SOA to improve software development process, Tékhne, 2008, Vol VI, no 9, ISSN: 1645-9911
[17] Vos, W., and Matthee, M.C., 2011, Towards A Service-Oriented Architecture: A Framework For The Design Of Financial Trading Applications In The South African Investment Banking Environment, South African Journal of Industrial Engineering May 2011 Vol 22(1)

Dr. Wiranto Herry Utomo is an associate professor of the Departement of Information System at the Satya Wacana Christian University, Salatiga, Central Java, Indonesia. He has published in several journals including IJWA, MASAUM Journals, and international conference including iiWAS of the ACM. His research interests include the enterprise integration, strategic alignment, SOA, Web Services, BPEL, Enterprise Service Bus, and Java EE.