Journal of Theoretical and Applied Information Technology
31st March 2015. Vol.73 No.3
© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

GENERIC ANALYSIS METAMODEL BASED ON SERVICE ORIENTED SOFTWARE DESIGN

NIK MARSYAHARIANI NIK DAUD[1], WAN M.N. WAN KADIR[2]
Software Engineering Department,
Faculty of Computing
Universiti Teknologi Malaysia,
81310 UTM Johor Bahru
Malaysia
E-mail: [1]mndnik2@live.utm.my, [2]wnasir@utm.my

ABSTRACT

Analyzing software in order to evaluate its quality attributes is not a foreign concept in software lifecycle. There are many methods that can be implemented to analyze software with some focusing on source code analysis and others concentrating on analyzing software artifacts from various phases such as design artifacts. Service oriented software is not excluded from this. As software that uses services as part of its functionalities, service oriented software designs represent more completed view of the software compared to other artifacts. This work focus on identifying elements from design artifacts that represent service oriented software structure and behavior based on standard modeling language for service oriented architecture. The elements are then represented in a generic metamodel that can be used to analyze the software. A case study will be used to show how design artifacts from selected standard modeling language, SoaML being mapped to model based on proposed metamodel.

Keywords: *Service oriented software, metamodel, modeling language, SoaML*

1. INTRODUCTION

Developing software that implements services as part of its functionalities is not uncommon these days. The introduction of service oriented architecture (SOA) into software development world had encouraged the reusability of existing business functionalities by repackaging it into services. These services are exposed to any interested party that uses it to develop software. These software, also known as service oriented software (SOS), can either be software that partially implements services as part of software or it could be a collection of services that work together to fulfill certain business goals.

SOS bears some similarities with previous development approach namely component based software, in that both enforce on abstraction on its implementation by encapsulating the implementation. The concept of abstraction realized loose coupling in between elements in software thus making it better in quality. SOS however differs from component based software in that components usually belong to one organization while services in software may belong to different organizations. Services usually offered by different organizations and can be used by a diversity type of clients thus services usually technology agnostic and support location transparency [1]. Implementing SOA in software take the abstraction to another level as services are closely related to business compare to component based that is more on technical level. Services are considered more as realization of business functionality while component based is created from the implementation modularization view.

These dissimilarities bring new problems in developing software. For example, having distributive ownerships for services implemented in software is making analyzing software more complicated as one cannot perform code analysis due to copyright issues and code absence. Analysis on SOS however can be done using other artifacts such as design artifacts. Elements in SOS and its relationships can be studied based on design artifacts of SOS. Apart from this, relation between

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

services and business functionalities need to be considered while analyzing SOS.

Analyzing software is a common process that can be adapt in any phases in software lifecycle. Inputs for software analysis are usually taken from software artifacts belonging to different phases of software lifetime, starting from requirement phase up to maintenance. On the early age of software development, analysis is done mainly based on program code though in recent years, due to diversity of software development techniques such as model driven architecture (MDA), other artifacts such as document specification and design artifacts has also been used as input of software analysis[2].

To analyze SOS based on design artifacts is challenging as there are few modeling language proposed to design SOS. Each modeling languages cater specific area in SOS. To come out with design artifacts that can be used to analyze quality of SOS, a generic metamodel is proposed. Metamodel is one of concepts introduced in MDA that can be defined as model used to describe model. This paper will discuss on the proposed generic metamodel.

This paper is structured as followed: Section 2 provides background on SOS design and metamodel. Apart from that, selected SOS modeling language will be briefly discussed. Section 3 discusses on elements available in SOA design artifacts. Further explanation on the chosen key elements from SOS development perspectives will be inserted. Section 4 presents the proposed generic metamodel that is built based on selected key elements and its relationships. This is followed by application of the metamodel using a simple travel agency reservation system as a case study. Finally, Section 5 concludes this paper.

2. BACKGROUND

Software development processes are geared up towards using model as it main artifact by promoting model as at par with codes. This innovation is especially accelerated with the introduction of MDA, one of OMG initiatives[3]. In MDA approach, models are used as enabler to communicate between stakeholders in understanding software. One of the main concepts introduce in MDA is metamodel. Metamodel can be described as model that describes modeling and is derived from Meta Object Facility (MOF), a language used to describe metamodel. Metamodel can be used as tool in describing specific domain[4-5]. The relationships between metamodel and its predecessor and successor can be seen in figure 1.

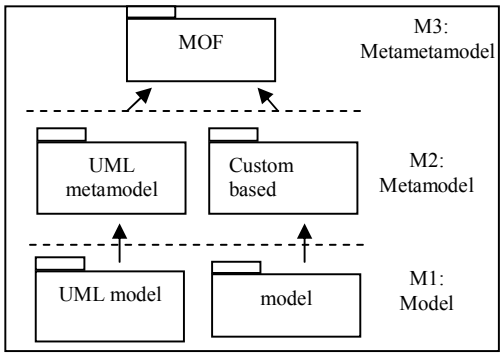

Figure 1: MDA metamodel Layer

There are 4 layers in metamodeling where M0 is excluded from figure 1 as it represented real world. M1 represents the usual model that has been used to model real world such as UML model or any domain specific model. Metamodel resides in M2 layer where models in this layer are used to describe model. For example, to use UML model, there are certain syntax and semantics that user need to know in order to use UML model. This syntax and semantics is described using UML metamodel that in turn are described using MOF, a metametamodel which resides in M3.

Other metamodel can be created to represent a domain specific model. This is known as custom based metamodel. Apart from using metamodel, UML had a special flexibility known as UML profile. Using UML profile, one can extend UML metamodel by using stereotype that enabled user to use UML model for more specific domain. The advantage of using UML profile over domain specific metamodel is that user can used existing tool for UML to create their model. Using both approaches, many researchers had proposed metamodel for SOA.

In SOA related researches, metamodel has been used as tool to describe elements and relationships of the area [6-8]. Each metamodel differs based on researchers point of interest. For example, metamodel is used to define on privacy policy in SOA [9] and fault tolerant of SOS [10]. On the other hand, more generic metamodel had also been proposed such as SoaML by OMG[11]. SoaML or SOA Modeling has been accepted as standard with its specification releases on 2012.

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PIM4SOA [12] is another example of metamodel that had been proposed as a generic model . PIM4SOA or Platform Independent Model for SOA is proposed to support MDA approach. The metamodel focuses on modeling business processes' interactions. What makes it different from other metamodel is that PIM4SOA focuses on four aspects which are service, process, information and quality of service[12]. This metamodel has been implemented as part of plug-ins for *sourceforge* tool. Another attempt to model SOA at PIM level is done by [13].

Service Component Architecture (SCA) is a collection of specifications published by OASIS group in 2005 to cater on development of SOS. SCA can be divided into four major models and for this paper the scope will be on SCA Assembly model [14]. SCA Assembly model specify on structure encompasses of components and how they are linked to each other. It is supposed to be technology agnostic though most of SCA model elements are materialized as XML implementation.

Service Reference Modeling Language (SRML) is a modeling language proposed from SENSORIA project, a collaboration project between universities in Europe ended on 2010. It is a technology agnostic language which specifically focuses on service excluding middleware infrastructure of SOA[15] . SRML supports its syntax structure using formal semantic thus its documentation focuses more on this aspect.

The selection of these modeling languages is by no mean trying to ignore others proposed works. These modeling languages are selected as references on what elements of SOS are commons in these SOS designs. Though there are many metamodel proposed for development of SOS, each cover different aspects of SOS thus a generic metamodel is proposed. The generic metamodel is specifically being used for analyzing quality in SOS.

3. KEY ELEMENTS IN SERVICE ORIENTED SOFTWARE DESIGN

Software design can be viewed from two perspectives that are architectural design and low level design. Architectural design perspective deals with software by viewing components and connections in software. Components reflect elements available in software where the relationships and structure of these components are shown based on connections. Interactions between components to describe on how a software achieves its' functionalities are emphasized at architectural design level. As opposed to this, low level design operates on implementation design structures. Details on operations and parameters are such examples on the focus of low level design as this level is one step closer to software's implementation.

Going through this, most of modeling languages cover both levels but for this work, the focus would be on architectural level design. To analyze on software structure and behavior, the level of interactions and dependencies between components in software are important segments to be studied. Architectural level design covers both segments of software thus it is already sufficed for software's analysis.

Works in SOS modeling have branched into many specific areas especially with the rising of web services technology, attracting researchers to focus their works in this area. Some have been working on more general level without being too dependent on any technology. Modeling SOS is different from other area in SOA as it usually ignores other services' infrastructures such as service middleware and service registry. This is due to the fact that SOS is viewed from developers' perspective and the mechanism on how services are published and delivered are not their main concerns. It is more on how services interact with components of software and how can it be removed or added dynamically during runtime without disrupting software's functionalities is more important for developers.

Most service modeling design supports this idea, generally. SoaML, a modeling language which implements UML Profile as extension from UML 2.0, includes stereotypes such as participant and component diagram to model components' interactions and structures. SCA assembly model, an initiative from open OASIS, is used to model services from service component aspects. The same goes to SRML, a language proposed by research collaboration named SENSORIA, where SRML Module structure is used to model components' architecture of a service. As a support for MDA approach, a modeling language specifically for SOA at platform independent level, PIM4SOA has been developed. Using these four modeling language as main references, key elements for service component is identified as can be seen in Table 1.

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Analysis on current modeling languages is done based on structure and behavior of software. Elements that are identified to represent SOS's structures are component, service, service interface and connection between these elements. The connections are divided into two types, interconnection representing connections between elements at architecture level and intra-connection representing connections within composite element be it service or component. Behavior analysis of SOS is done from static interaction and dynamic interaction. This is necessary as one of the unique attributes for SOS is the capability for the software to add and remove any services during runtime thus reflecting a dynamic interaction.

Study had been done on four aforementioned service modeling languages that cover SOS generally without being too specific in certain aspects. The study covers on both structure and behavior that may represent service modeling. As described in Table 1, four key elements in SOS had been selected which are component, interface, connection and service. Each of them is further divided into two subcategories except for component.

The component element exists in three selected service modeling except for PIM4SOA. For interface element there are two types of interfaces that need to be considered. Those are served interface and required interface. Served interface acts as access to service for user while required interface is includes when a service required certain operations from other elements in order to fulfill it task. For SoaML, service port and required port are considered as interface as these elements are typed by interface either it is service interface element or simply an interface. Both SCA and SRML include served and required interfaces. PIM4SOA had an element called endpoint that can be interpreted as access for service usage.

The next element in SOS to be considered is service. In service modeling, service is treated as abstract concept where most of the time it is realized by other concepts. In this case, in each service modeling language, concepts that are considered to realize service are chosen. Service are categorized into two categories, atomic and composite, where atomic service represents the simpler version of service that contains only implementation and interface whilst composite service covers more complex service that contains services interaction with each other to achieve certain business functionality. A pure SOS can also be seen as a composite service. Each modeling language has its own concept that can be translated into both atomic and composite service as can be seen in Table 1.

The last element, connection element is used as connectors between other elements in SOS. Connections are divided into interconnection and intra-connection for analysis purpose. Interconnection is connection used between services in composite service and at software level. This is treated as analysis from architectural level perspectives. Intra-connection is used to connect internal elements in a service such as interface and component. PIM4SOA is the only modeling that didn't mentioned on connections as it uses messages for elements' interactions.

From behavior of SOS view, this study shows that behaviors in SoaML, SCA and SRML are usually represented using behavioral diagrams from UML 2.0. Apart from this, BPEL files are also used to represent workflow of business process activity. However, runtime behavior such as adding and removing services are rarely mentioned. Thus, in this study, these two behaviors will be included into proposed metamodel. Table 2 summarizes on behavior included in these modeling languages.

It can be summarized that component, interface, service and connection are key elements needed in representing SOS as it presents in all three service modeling language except for PIM4SOA. This might be attributed by the fact that PIM4SOA tends to represent SOS at a very high level as compared to others modeling languages. Thus it didn't cover on low level design aspects. Based on this study, a generic metamodel is proposed using the elements. The metamodel is discussed in next section.

4. GENERIC ANALYSIS METAMODEL

Previous section discussed on the extraction of key elements from design artifacts based on existing service modeling languages. Using these identified elements, a metamodel is proposed specifically for quality analysis based on structures and behaviors of SOS. Metamodel technique is elected to represent existing elements its relationship due to several factors. Metamodel allows for easy mapping between models based on MDA approach. The proposed generic metamodel is shown in figure 2.

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

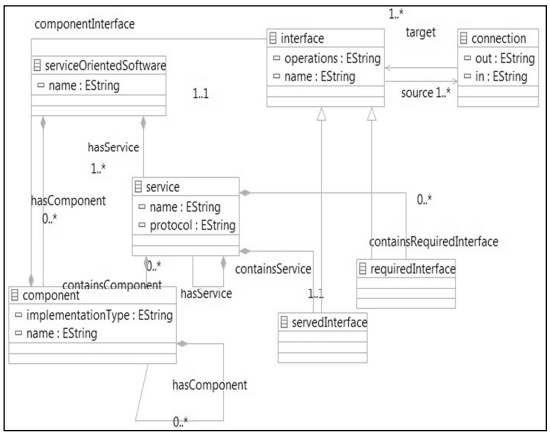

Figure 2: Generic Metamodel

This metamodel is generated using Eclipse Modeling Tool. Each key element is represented in this metamodel where relationships between these elements are elaborated. The structure of this metamodel is divided into two main models that are service oriented software model and service model. Service oriented software model represents elements that exist in architecture level of the software, emphasizing on connections between services in order to fulfill a business goal while service model represents elements at low level design.

From architecture perspective, there are four main elements in this model. The elements are component, service, interface and connection. Component and service represent elements that implement business functionalities in SOS.

The first rule for elements in SOS is that SOS must at least contain one instance for each element. Component element has implementationType attribute that store the value of implementation type for the component. Examples of the type are .java or .bpel. Service element has two attributes which are description and protocol. Description stores information about the service while protocol store name of file that elaborates on behavior of the service. Connection is used to define connection between components and services where the interactions for service are represented using interface.

The service model contains four elements which are component, service, interface and connection. The second rule for elements is that service must at least have one instance of component, interface and intra-connection. A service must at least have one served interface and may not have any required interface at all. All connections in SOS are required to be done through interface elements. Two service specific elements, served interface and required interface inherited from interface element.

This generic metamodel focuses more on relationships in SOS and elements exist in it. This is due to the fact that this metamodel is generated for quality analyzing purpose. Behavior of SOS is not included as part of the metamodel as it will be represented using graph transformation system approach. The discussion on behavior of SOS will not be included in this paper.

Table 1: Key Elements in Service Modeling from Structural Aspect

Service modeling language	**Feature (Structure)**						
	Component	***Interface***		***Service***		***Connection***	
		Served	***Required***	***Atomic***	***composite***	***Inter-connection***	***Intra-connection***
SoaML	Component	Service port	Request port	Participant	Services Architecture	Service channel	Service channel
SCA Assembly model	Component	Service	Reference	Service component	Composite	Wire	Promotion
SRML Module Structure	Node	Serves-interface	Requires interfaces	Service module	Activity module	Wire interface	connector
PIM4SOA	-	Endpoint	-	Collaboration	Collaboration	-	-

Journal of Theoretical and Applied Information Technology
31st March 2015. Vol.73 No.3
© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: Key Elements in Service Modeling from Behavioral Aspect

Service modeling language	Feature (Behavior)					
	Dynamic		*Static*			
	Add service	*Remove service*	*Invoke*	*Reply*	*Receive*	*Close*
SoaML	-	-	✓	✓	✓	✓
SCA Assembly model	-	-	✓	✓	✓	✓
SRML Module Structure	✓	✓	✓	✓	✓	✓
PIM4SOA	✓	✓	✓	✓	✓	✓

5. CASE STUDY

The proposed Generic Metamodel is applied on a case study of Travel Agency Booking System. This case study is selected as it shows simple interactions between services in order to fulfill certain business goal which in this case is making reservation for travel agency. The system consists of 4 main services; travel agency system, hotel booking services, transportation component and flight booking services. Travel agency system acts as the main component in this software by connecting to three other external services in order to book hotel, flight and transportation for the travel agency's customers. These three services will return confirmation on whether the reservation is succeeded.

To model this case study, SoaML is chosen as modeling language where selected diagrams will be produced to represent the case study. The selected diagrams servicesArchitecture diagram and participant diagram, and component diagram. servicesArchitectures diagram represents the interaction between participants at architecture level and participant diagram shows on the structure at service component level. Component diagram simply represent component available in the software. These three diagrams will be used as inputs to come out with its equivalence model based on proposed metamodel.

Implementation of this case study is done using Eclipse Modeling Tool. Eclipse is an open source framework that enabled user to create adds-on to its framework. It also enables the use of ATL (Atlas Transformation Language) that is used as transformation language for this case. Transforming sample model can be explained based on the following Figure 3. Basically, a sample model based on SoaML metamodel will be generated and will be used as input for a transformation program using ATL. The output will be a model that is compliance to generic metamodel.

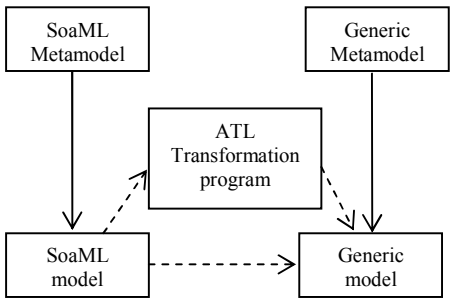

Figure 3: Transformation model

A sample of the SoaML model can be seen from Figure 4. The sample is in XMI filetype with servicesArchitecture as the main model. As depicted in SoaML metamodel, a servicesArchitecture contains participant and serviceContract. Some of the participant is shown in the Figure 4, for example the "travel agency" participant. This figure only shows part of the whole model. A transformation program is developed based on mapping rules. For example, servicesArchitecture from SoaML model is mapped into service oriented software in generic metamodel. The result is automatically generated and portion of the result model can be seen in Figure 5.

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
<xmi:XMI xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI" xmlns="soaml">
<servicesArchitecture name="travelBookingSystem">
    <containsParticipant name="travel agency">
        <containsComponent name="bookingModule"
type="java">
            <hasServicePort>
                <typedBy interfaceName="mainInterface"
operation="bookTransportInterface"/>
            </hasServicePort>
        </containsComponent>
        <containsComponent name="transportQuery"
type="java">
            <hasRequestPort>
                <typedBy interfaceName="transportInterface"
operation="bookTransportInterface"/>
            </hasRequestPort>
        </containsComponent>
        <containsComponent name="flightQuery" type="java">
```

Figure 4: Sample of input model

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <xmi:XMI xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:genMet="genMet">
    <genMet:serviceOrientedSoftware
name="travelBookingSystem">
    <hasComponent name="travel agency">
    <hasComponent implementationType="java"
name="bookingModule"/>
    <hasComponent implementationType="java"
name="transportQuery"/>
    <hasComponent implementationType="java"
name="flightQuery"/>
    <hasComponent implementationType="java"
name="hotelQuery"/>
    </hasComponent>
    <hasComponent name="transport">
    <hasComponent implementationType="java"
```

Figure 5: Sample of output model

6. CONCLUSIONS AND FUTURE WORKS

Software design can be represented by multi type of designs. Even for SOS which is a new type of software, many works have been proposed regarding on modeling its' design. This work proposes a refined version of current design artifacts that generalize elements in SOS modeling. A study had been made on standard, published works on modeling language for SOA such as SoaML. Key elements exist in the modeling language are identified. Based on the elements, a generic metamodel is proposed and presented in this paper. A simple case study on travel booking system is chosen to show on how the proposed metamodel can be transformed from selected service modeling language, in this case, SoaML. The case study shows that the generic metamodel managed to refine existing model, by representing key elements of the models.

This paper focuses only on structural aspect of generic metamodel for SOS. Behavioral aspect is out of this paper scope. Future works will include on showing the representation of behavioral aspect and analyzing the metamodel from quality perspectives. It is hoped that this work will help in SOS developer to analyze their software behavior.

ACKNOWLEDGEMENT

The authors would like to express their deepest gratitude to Research Management Center (RMC), Universiti Teknologi Malaysia (UTM) and Ministry of Education Malaysia for their financial support under Fundamental Research Grant Scheme (Vot number R.J130000.7828.4F216).

REFERENCES

[1] M. Papazoglou, "Service-oriented computing: Concepts, characteristics and directions," in *WISE: Web Information Systems Engineering, 2003*, 2003, pp. 3-12.

[2] D. Jackson and M. Rinard, "Software analysis: a roadmap," presented at the Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland, 2000.

[3] S. J. Mellor, *et al.*, *MDA Distilled: Principles of Model-Driven Architecture* Boston: Addison-Wesley, 2004.

[4] C. Atkinson and T. Kühne, ""The Role of Metamodeling in MDA"," in *International Workshop in Software Model Engineering*, Dresden, Germany, October 2002.

[5] R. Gjataj, "Metamodel Based Editor for Service Oriented Architecture " Master, Department of informatics, University of Oslo, 2007.

[6] L. Baresi, *et al.*, "Style-based modeling and refinement of service-oriented architectures," *Software & Systems Modeling,* vol. 5, pp. 187-207, 2006/06/01 2006.

[7] N. A. Delessy and E. B. Fernandez, "A Pattern-Driven Security Process for SOA Applications," in *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, 2008, pp. 416-421.

[8] C. Marin, *et al.*, "A MDE Approach for Power Distribution Service Development," in *Service-Oriented Computing - ICSOC 2005*. vol. 3826, B. Benatallah, *et al.*, Eds., ed: Springer Berlin Heidelberg, 2005, pp. 552-557.

[9] D. S. Allison, *et al.*, "Metamodel for privacy policies within SOA," in *Software Engineering for Secure Systems, 2009.*

© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SESS '09. ICSE Workshop on, 2009, pp. 40-46.

[10] F. Mahdian, *et al.*, "Modeling Fault Tolerant Services in Service-Oriented Architecture," in *Theoretical Aspects of Software Engineering, 2009. TASE 2009. Third IEEE International Symposium on*, 2009, pp. 319-320.

[11] Object Management Group. (2012, 30 October 2013). *Service Oriented Architecture Modeling Language*. Available: http://www.omg.org/spec/SoaML/1.0.1/

[12] G. Benguria, *et al.*, "A platform independent model for service oriented architectures," in *Enterprise interoperability*, ed: Springer, 2007, pp. 23-32.

[13] M. López-Sanz, *et al.*, "Modelling of Service-Oriented Architectures with UML," *Electronic Notes in Theoretical Computer Science,* vol. 194, pp. 23-37, 2008.

[14] M. Beisiegel, *et al.* (2011, 31 October 2013). *Service component Architecture Assembly Model Specification*. Available: https://www.oasis-open.org/committees/download.php/42321/sca-assembly-1.2-spec-wd04.pdf

[15] J. Abreu and J. Fiadeiro, "A Coordination Model for Service-Oriented Interactions," in *Coordination Models and Languages*. vol. 5052, D. Lea and G. Zavattaro, Eds., ed: Springer Berlin Heidelberg, 2008, pp. 1-16.