An MDA-Based Modeling and Design of Service Oriented Architecture

Adel Torkaman Rahmani, Vahid Rafe, Saeed Sedighian, and Amin Abbaspour

Iran University of Science and Technology
Computer Engineering Department
Tehran, Iran
{rahmani, rafe, sedighian, abbaspour}@iust.ac.ir

Abstract. Traditional approaches to software systems development such as using tools and modeling frameworks are appropriate for building individual object oriented or component based software. However they are not suitable for designing of flexible distributed enterprise systems and open environments. In recent years, service-oriented architecture (SOA) has been proposed as a suitable architecture for development of such systems. Most current approaches in employing SOA are tailored to specific domains and hence are not general purpose. Therefore, in order to gain the full benefits of such technology, a more effective general approach to modeling and designing these complex distributed systems is required. In this paper, we present a model-driven approach to SOA modeling and designing complex distributed systems. In this approach, first the PIM of the business system is derived and expressed in standard UML modeling constructs and then this PIM is transformed to the SOA-based PIM by some transforming tool. After the SOA-based PIM is obtained, it can be used to generate PSM for a specific platform such as Web Services, Jini or other platforms. To make it clear how this PSM could be generated we will use Web Services as a target platform and the steps of this transformation will be shown.

1 Introduction

Traditional approaches to development of software systems such as using tools and modeling frameworks are appropriate for building individual object oriented or component-based software. But they are not suitable for designing flexible distributed enterprise systems, the systems that are more complex than before. In fact, the design and implementation of modern software system is complex and costly due to rapid and continuous technology changes. They are complex not only for the volume of code and data involved is much larger but also for many other reasons such as integrating new aspects in those systems (e.g. security, reliability and performance) [1].

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. And hence a new paradigm like service-oriented architecture [2] is necessary.

SOA is a paradigm that utilizes services as fundamental elements for developing applications [3]. 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.

This paper presents a model-driven approach to SOA modeling and designing complex distributed systems based on MDA. A new paradigm called Model-Driven Architecture (MDA) has been proposed to develop of complex distributed systems [4]. 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.

The paper is organized as follows: section 2 explains a brief overview of MDA and SOA. In section 3, an illustrative example is introduced through the related use case diagram. The proposed approach is demonstrated in section 4. In this section the PIM of the MCRI system created with UML will be shown. The steps of transformation from this PIM to PSM based on Web Services will be illustrated. Last but not least, section 5 will conclude the paper.

2 Background

Whereas our research is founded on MDA and SOA approaches, a brief overview of what is aimed by these two approaches is necessary.

2.1 MDA

The OMG's Model Driven Architecture (MDA) separates the modeling task of the implementation details, with out losing the integration of the model and the development in a target platform [5]. The key technologies of MDA are Unified Modeling Language (UML) [6], Meta-Object Facility (MOF), XML Meta-Data Interchange (XMI) [7] and Common Warehouse Metamodel (CWM).

The core paradigm of MDA is model transformation. With MDA, system construction consists of a sequence of models and transformations between these models. The model-driven approach starts with a platform-independent model (PIM), which is subsequently transformed into a platform-specific model (PSM). The PSM is then transformed to code. Generally, PIM and PSM are defined using OMG's Unified Modeling Language (UML) [8].

The main idea of MDA is a pattern: translating from PIM (Platform Independent Model) to PSM (Platform Specific Model) [9]. In theory PIM is a model that is independent from any platform. But for making things actual, we need a final code that works on a specific platform. Using translation process in MDA, we can make some

PSMs for different platforms. Then PSM can be translated to actual code for execution. If in future new platforms or frameworks arise, we can translate our unchanged PIM to these new platforms. MDA pattern can be applied more. In this mechanism that proposed in MDA Guide [10], PSM of Level 1 is PIM for Level 2, etc.

2.2 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 [12].

3 Illustrative Example

To study the issues of our approach, we have chosen an illustrative example of a Management system which manages some Customer Relationship Information (MCRI).

MCRI system receives queries from customers and sends them the relationship information that is required by the query. Figure 1 presents a simplified use case diagram for a MCRI system.

Figure 1 shows a customer who needs to make a contact. In order to access the contact information, he/she makes a conversation with an employee in the MCRI system. If the customer does not receive his/her requirements then he/she makes a request for appointment and the employee replies the customer's request.

In the next section, after presenting of our proposed architecture, this illustrative example will be modeled and designed based on our proposed approach.

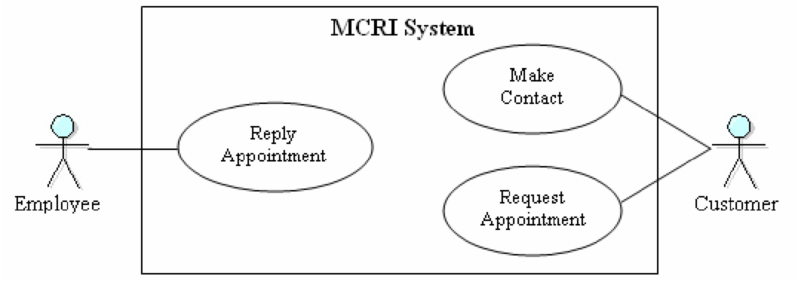


Fig. 1. Use case Diagram of the MCRI System

4 Our Proposed Approach

Modeling service-oriented solutions for above-mentioned systems is not a straightforward task. Due to the complexity of service-oriented solutions (such as defining the scope of services, elements of each service …) and the diversity of available technology platforms, the MDA approach to designing SOA seems a natural choice. Thus we propose an MDA-based approach, whose framework is shown in figure 2.

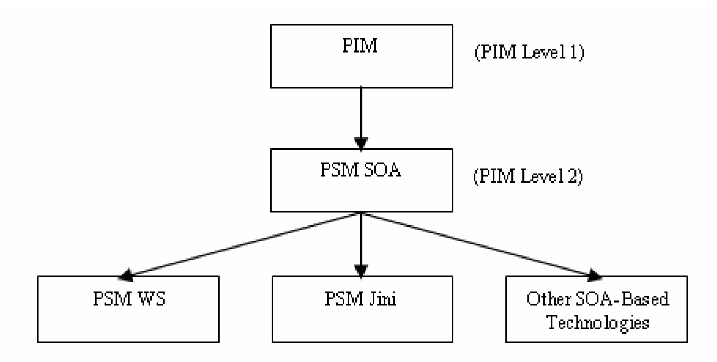


Fig. 2. Proposed Architecture

In this framework, the PIM of the system is created using UML diagrams by the analyst of the system. He/She can analyze the system without worrying about identifying the services and their properties such as scope, loosely-coupling, granularity and so on. Therefore PIM of the system will be designed simply without thinking about services. This is the benefit of the MDA approach.

This phase, designing of PIM, is pretty simple and is accomplished like component-based systems. But what in our proposed approach is important, is how 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 component-based systems the patterns which are used to determine the PSM of the system have a specific form. For example, in many of these patterns a ‘set’ and a ‘get’ method is added to the PSM for each attribute in PIM class diagrams. Because of differences between implementing services (in service-oriented systems) and implementing components (in component-based systems), such methods in component based systems are useful but in service oriented systems these methods have a high communication cost. For example, in component-based systems each component is situated in a container which is responsible for managing all the instances of components. In contrast, for each service there is a single instance which manages a set of resources and consequently, unlike components, services are for the most part stateless [13]. This means that we need to view a service as a manager object that can create and manage instances of a type, or set of types. This yields a design pattern that makes use of value objects which represent the instance states, for systems that need to maintain and use these instant states. This method of passing the state of a service to its client implies that rather than using a large number of small operations to retrieve the service state, only a single large operation is required. This has enormous impacts on network usage for remote services.

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 fact this second PIM is a PSM.

In this transformation, for each class diagram in PIM, 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, Jini and/or other platforms with transforming tools.

In the next subsections, the illustrative example will be modeled and the PIM based on SOA will be generated using our approach and then this model will be transformed to the PSM based on Web Services (as a chosen platform).

4.1 The PIM of the MCRI System

The PIM of the illustrative example, is showed in figure 3 in this paper, the fragment of the PIM is presented with out other modeling part to simplify the presentation. As shown in figure 3, the MCRI system consists of three classes that implement the main characteristics of the system: *Account*, *Contact* and *Appointment*.

The customer, accesses the *Account* to make a *Contact*, and if it is required then he/she sets an *Appointment*.

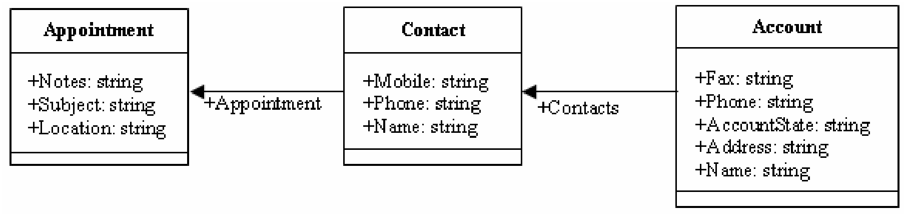
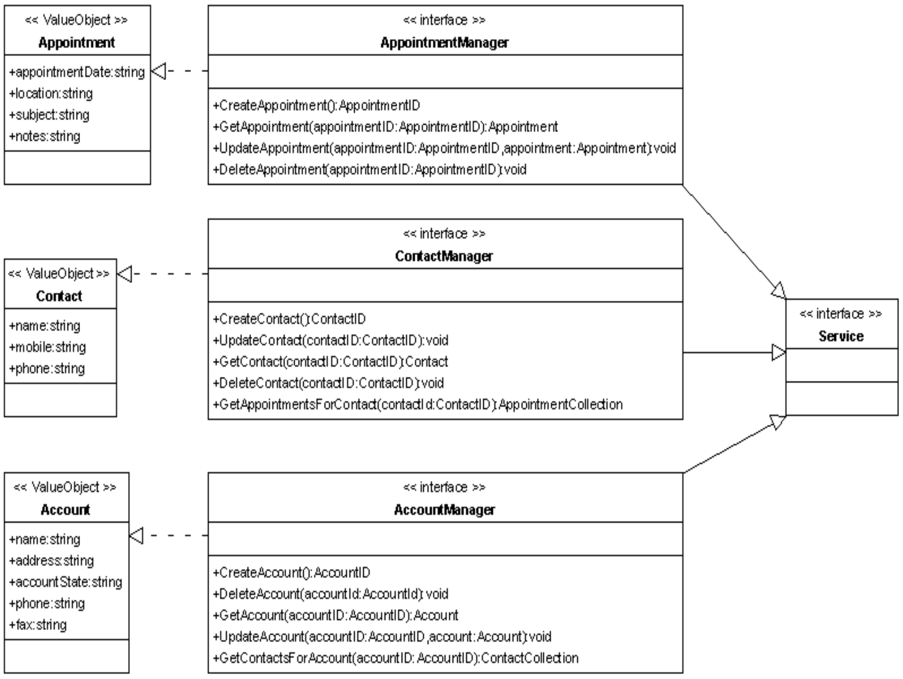


Fig. 3. The PIM of MCRI system

4.2 Generating the PIM Based on SOA

As we mentioned at the beginning of this section, for transforming the PIM to the PSM based on SOA we use a special pattern. As it is shown in figure 4, for the MCRI system three service managers have been designed (AppointmentManager, ContactManager, AccountManager). Each of these service managers has a value object that manages it. For each value object, the following methods are added in the related service manager: Create, Delete, Get and Update. Such as CreateAppointment, DeleteAppointment and … in AppointmentManager.



<< ValueObject >>
Appointment
+appointmentDate:string
+location:string
+subject:string
+notes:string
<< interface >>
AppointmentManager
+CreateAppointment():AppointmentID
+GetAppointment(appointmentID:AppointmentID):Appointment
+UpdateAppointment(appointmentID:AppointmentID,appointment:Appointment):void
+DeleteAppointment(appointmentID:AppointmentID):void
<< ValueObject >>
Contact
+name:string
+mobile:string
+phone:string
<< interface >>
ContactManager
+CreateContact():ContactID
+UpdateContact(contactID:ContactID):void
+GetContact(contactID:ContactID):Contact
+DeleteContact(contactID:ContactID):void
+GetAppointmentsForContact(contactId:ContactID):AppointmentCollection
<< interface >>
Service
<< ValueObject >>
Account
+name:string
+address:string
+accountState:string
+phone:string
+fax:string
<< interface >>
AccountManager
+CreateAccount():AccountID
+DeleteAccount(accountId:AccountId):void
+GetAccount(accountID:AccountID):Account
+UpdateAccount(accountID:AccountID,account:Account):void
+GetContactsForAccount(accountID:AccountID):ContactCollection

Fig. 4. Generated Service-Oriented Architecture

For each association in the PIM, one or more methods are added to the service manager. For example, GetAppointmentForContact in the ContactManager.

4.3 Generating the PSM Based on Web Services Platform

After generating PSM based on SOA, the final PSM will be designed using a target platform such as Web Services or Jini or …. In our example, we use Web Services as a target platform.

Web Services are defined by WSDL (Web Services Description Language) [14]. WSDL is an XML-formatted language used to describe a Web service.

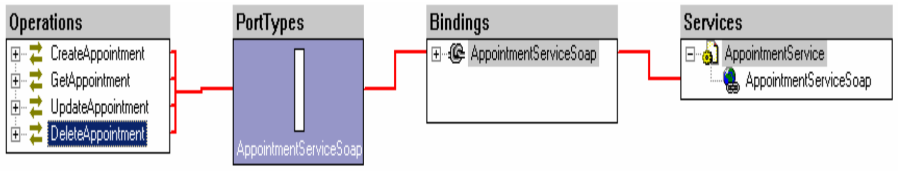
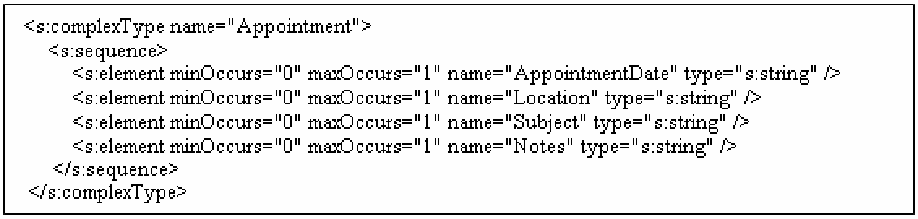


Fig. 5. A part of generated PSM besed on WS for AppointmentManager

Transforming PSM based on SOA to the PSM based on Web 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. For example, a part of generated PSM based on Web Services for AppointmentManager is shown in figure 5.

And its generated WSDL is shown in figure 6. This figure presents the definition of Appointment.



```
<s:complexType name="Appointment">
   <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="AppointmentDate" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="Location" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="Subject" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="Notes" type="s:string" />
   </s:sequence>
</s:complexType>
```

Fig. 6. Definition of Appointment using WSDL

5 Conclusion

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 (such as Web Services, Jini and so on) 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, EDOC 2004.
2. Michael N.Huhns, Munindar P.Singh, Service-Oriented Computing: Key Concepts and Principles. Journal of IEEE Internet Computing, 2005.
3. Mike P.Papazoghlou, Service-Oriented Computing: Concepts, Characteristics and Directions. Proceedings of the Fourth international Conference on Web Information systems Engineering, 2003.
4. S.Cook. Domain-Specification Modeling and Model Driven Architecture. MDA Journal, Pages 1-10, 2004.
5. OMG. Model Driven Architecture (MDA) - document number ormsc/2001-07-01, 2001.
6. Object Management Group. Unified Modeling Language: Superstructure, Apr 2003. Document number: adi2003-04-01.
7. Object Management Group, XML Metadata Interchange (XMI) specification, May 2003. Version 2.0, formal/03-05-02.

8. Markus Debusmann, Reinhold Kroeger, Fachhochschule Wiesbaden, Unifying Service Level Management using an MDA-based Approach , proceedings of the 2004 IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, South Korea, 19.-23. April 2004.
9. Model Driven Architecture – Applying MDA to Enterprise Computing, David S. Frankel – Wiley Publishing, Inc (OMG Press) – 2003
10. Object Management Group. MDA Guide, V1.0.1, omg/o3-06-01, June 2003.
11. Hao He, What is Service-Oriented Architecture?, Orielly WebService Site. http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html, September 2003
12. Qusay H. Mahmoud, Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI), Sun Developers Network, Sun Developers etwork, April 2005
13. Zoran Stojanovic, Ajantha Dahanayake, Henk Sol, Modeling and Design of Service-Oriented Architecture, in proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics, October 10-13, Netherland.
14. E. Christensen, F. Curbera, G. Meredith and S.Weerawarana, Web Services Description Language (WSDL) V.1.1, 2001.