Design of NGOSS TSA Using Web Services Technologies

Mi-Jung Choi
School of Computer Science
University of Waterloo
Waterloo, Canada
mjchoi@cs.uwaterloo.ca

James Won-Ki Hong
Dept. of Computer Science and Engineering
POSTECH
Pohang. Korea
jwkhong@postech.ac.kr

Hong-Taek Ju
Dept. of Computer Science
Keimyung University
Daegu, Korea
juht@kmu.ac.kr

Dong-Sik Yun
KT Network Technologies Labs
Korea Telecom (KT)
Daejeon. Korea
dsyun@kt.co.kr

***Abstract* — To reduce frequent changes and upgrades of management systems, we need a guideline of OSS's architecture and development methods of the OSSs. TMF has proposed NGOSS technology-neutral architecture (TNA) which describes major concepts and architectural details of the NGOSS architecture in a technologically neutral manner. The NGOSS TNA can be mapped onto appropriate technology-specific architectures (TSAs) using specific technologies such as XML, Java and CORBA. Web services, which is a distributed and services-oriented computing technology, can be applied to NGOSS TSA. In this paper, we examine the architectural requirements of TNA, and provide a design of Web services-based TSA in accordance with the TNA requirements.**

Keywords; NGOSS, TNA, TSA, Web services, WSDL, SOAP, UDDI, WS-BEPL

I. INTRODUCTION

Telecommunication infrastructures from multiple vendors deliver new and frequent changing combinations of services over every type of communication technology. Also, with the convergence of data communications, telecommunications, and other forms of communication, the complexity, heterogeneity, and size of networks supporting the emerging services are rapidly increasing. Therefore, we need flexible, scalable service fulfillment operations support system (OSS) solutions. For this reason, TeleManagement Forum (TMF) provides a Next Generation Operations Systems and Software (NGOSS) [1] framework to improve the management and operation of information and communication services.

As network services newly emerge, the network services and resources need to be easily configured. To configure the complex and heterogeneous networks and services, operation management systems also need to be developed using up-to-date technologies. To reduce frequent changes and upgrades of management systems, we need a guideline of OSS's architecture and development methods of the OSSs. Thus, TMF has proposed NGOSS technology-neutral architecture (TNA) [2], which describes major concepts and architectural details of the NGOSS architecture in a technologically neutral manner. The goal of TNA is to define a component-based and distributed system architecture and an associated critical set of system services which this architecture requires. The TNA can be implemented using the currently available distributed system information technologies. That is, the TNA can be mapped onto appropriate technology-specific architectures (TSAs), which can leverage industrial standard frameworks such as service-oriented architecture, component-based architecture and distributed computing. The separation of technology-neutral and technology-specific architectures enables OSS developers to choose the 'best fit' management components and technologies for their management capability.

TMF has also proposed three technology application notes that describe the mapping of TNA onto specific technologies such as XML [3], CORBA [4], and Java [5]. Web services is a distributed and services-oriented computing technology with strong support from the industry. It is one possible technology for NGOSS architecture and the mapping of NGOSS TNA to Web services-based TSA is a promising research area [6]. In this paper, we examine the essential parts of NGOSS TNA and propose a technology-specific architecture based on Web services technologies.

The organization of this paper is as follows. In Section II, we briefly investigate NGOSS TNA in the perspective of architecture and components, and Web services technologies as a related work. In Section III, we examine three technology application notes. We describe our design architecture in Section IV and present a case study in Section V. Finally, we conclude our work and discuss possible future work.

II. NGOSS TNA

In this section, we first briefly investigate the concepts of NGOSS TNA and its components. NGOSS is a set of guidelines and specifications for the industry to build software in a more structured and complementary way, based on

industry experience and previous and ongoing TMF activities. The NGOSS TNA [2] is the basic concept and component of NGOSS architecture. Figure 1 shows the detailed views of NGOSS TNA. The core components of TNA are common communications vehicle (CCV), services, and contract.

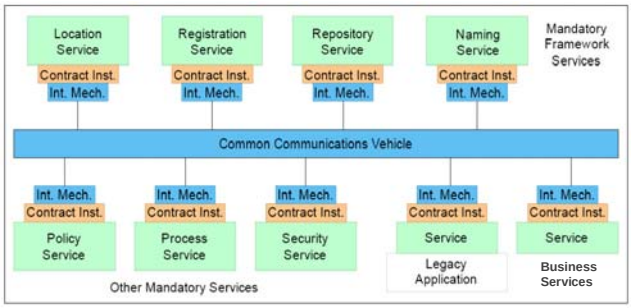

Figure 1. Detailed Views of NGOSS TNA

The service modules can communicate with each other through the common communications vehicle (CCV) [2]. The CCV is a kind of message bus independent of specific technology, which is responsible for the transport of information among application objects. The services are mainly divided into two parts: business services and framework services [2]. Business services provide the application level functionality that directly supports the implementation of a business process such as SLA management, billing mediation, QoS, etc. Framework services provide the infrastructure necessary to support the distributed nature of the NGOSS TNA. For example, they include naming and directory services, messaging services, and transaction management/monitoring services.

The framework services consist of repository, registration, location and naming services. The registration service provides for services and functionality needed to support location transparency. The repository service provides a logical view of all the information on a deployed distributed system. The naming service is responsible for generating and resolving unique names for the various entities contained in the repository. The location service, often built on the naming service, provides a means to map a request for an object to a particular instance of that object.

The NGOSS contract [2] is the fundamental unit of interoperability in an NGOSS system. The contract is used to define a specification of a service to be delivered, as well as to specify information and code that implement the service. In short, the contract is a way of reifying a specification of a service, and implementing the functionality of the service including obligations to other entities in the managed environment. Thus, it is much more than a container of data or a specification of a set of methods.

III. TECHNOLOGY APPLICATION NOTES

TMF provides three documents called application notes representing specific technologies applied to the NGOSS TNA. The application notes present requirements of the NGOSS TNA, an overview of each technology and a guideline to direct how the technology maps to the concepts of TNA. In this section, we present the principles of the NGOSS TNA and compare three TSA using specific technologies: XML, CORBA, and Java.

TMF proposed two technologies of XML [3] and CORBA [4] for NGOSS TNA and specified technology application notes with these technologies. OSS/J [5] initiative proposed Java technology as a specific technology for TNA. Table 1 shows a comparison result of three technologies in accordance with the alignment of NGOSS TNA.

Table 1: Comparison of Specific Technologies

Alignment	XML	CORBA	Java
Communication	HTTP, SOAP	GIOP, IIOP	RMI-IIOP, JMS
Framework services	▪ Runtime discovery service: UDDI ▪ Runtime location of information: XPath, XLink, XPointer	CORBA services (naming, trading, etc.)	▪ Runtime discovery service: JNDI
Business process management	Not specified	Define new services	Define business process component
Contract interface	XML definition	CORBA IDL	Java interface
Shared information	XML Schema (information adaptation: XSLT)	Can be defined by IDL	Not specified

XML has a natural affinity with communication management. The use of XML to validate, manipulate and define the structure of application specific information models becomes an attractive possibility. However, XML has a substantial overhead associated with the text-only encoding of messages [3].

CORBA is an open distributed object computing infrastructure. It automates many common network programming tasks such as object registration, location, and activation. Therefore, CORBA supports the communication method and framework services for the distributed processing. It also supports interface definition with specifying CORBA IDL. However, it does not provide information modeling; hence it can define the shared information using XML or other languages [4].

J2EE, Java implementation platform, directly implements the principles of NGOSS TNA such as the distribution support and separation of business process from implementation. However, Java does not have any explicit support for the concept of shared information or federated information services as defined in NGOSS TNA [5].

IV. DESIGN

In this section, we design a technology specific architecture using Web services technologies.

A. Alignments with NGOSS TNA using Web Services

We examine how the Web services technologies can be mapped onto the requirements and characteristics of the NGOSS TNA [6].

The information model is designed to be more than just a standard representation of data – it also defines semantics and behavior of, and interaction between, managed entities. The NGOSS Shared Information/Data (SID) model [7] provides the industry with a common vocabulary and set of information/data

definitions and relationships used in the definition of NGOSS architectures. The XML Schema allows us to define the structure of management information with richer definitions of data types of XML documents including complex and data centric types. The XML Schema also allows inheritance relationships between elements. Thus, we can define management information using XML Schema in a flexible manner. Based on our defined XML Schema, we can define management operations and messages written in WSDLs.

The second requirement of the NGOSS TNA is to provide distribution transparency. The NGOSS system needs a communication mechanism and a repository that records information used during system execution. This requirement is achieved by the CCV and framework services. UDDI provides a comprehensive mechanism for locating services at run time by storing service interfaces in their WSDL format. Application programs can search the UDDI repository to find the interface that they require, download the WSDL description and use the binding information to communicate with the interface over a suitable communication channel. UDDI supports basic functionalities of the framework services.

The requirement of separation of business process from software implementation is achieved by the definition of the contract and the process. The contract, which is a definition of a service specification and a message specification between service components, can be defined in WSDL based on the management information model in XML Schema format. The business process can be defined by WS-BPEL with the concept of process entities, process sequences and interaction messages. The WS-BPEL engine handles the process sequentially.

B. *Architecture*

Web service includes many standard specifications. In Section IV-A, we examined the alignment with NGOSS TNA using Web services technologies, in which we focus more on the NGOSS specification from a Web services perspective. Figure 2 shows our Web services-based TSA. Our architecture is extended from the TNA architecture using Web services technologies. As mentioned in Section IV-A, SOAP is used as CCV to communicate between process entities, and WSDL is used to define contracts between process entities through SOAP. UDDI supports the framework services of repository, registration, naming, and location services and WS-BPEL supports the process service.

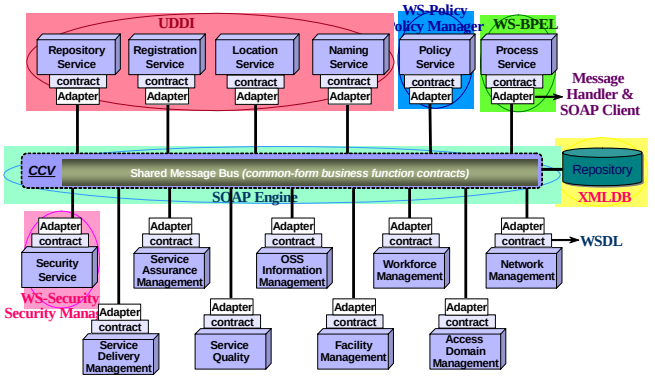

Figure 2. Web Services-based TSA

The policy service can use the WS-Policy specification as an information definition and a policy manager as an engine. The security service can use the WS-Security specification as an information definition and a security manager as a service engine. The management operation services such as service assurance, service delivery, etc. can be defined as new services using WSDL. The adapter acts as a message handler with a SOAP client module. A XMLDB can be used as a management repository instead of a traditional RDBMS. The native XMLDB stores the data structured as XML without translating the data to a relational or object database structure. However, the performance of XMLDB is not thoroughly verified yet.

V. CASE STUDY

We select the process of DSL fulfillment, which is one of the examples of eTOM's fulfillment process, as a sample case to verify our technology specific architecture.

A. *Management Information*

As explained in Section IV-A, we define our management information through XML Schema. We use the XML Schema defining the information model of customer order and service order. The information of customer order includes customer name, customer contract name, customer address, product, and so on. The service order contains all contents of the customer order that defines the information of the customer order as the complex type of CustomerOrderType and refers to this type. In this way, the customer order in the service order also refers to the same type so that it can be reused without any repeated definition.

B. *Management Process*

There are Order handling module, Service configuration & activation module, and Resource provisioning module in the management system for DSL fulfillment. The Order handling module receives the customer order request of a DSL service from a customer and sends the service order request to the Service configuration & activation module. The Service configuration & activation module asks the resource provisioning to the Resource provisioning module to acquire the resource after the service configuration.

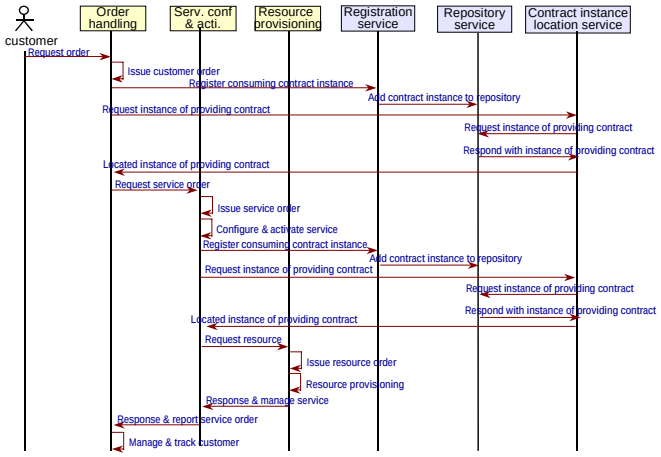

Figure 3. Management Scenario of DSL Fulfillment

Figure 3 is a combination of the interaction sequences between the framework service components and the sequence for the DSL fulfillment. That is, there should be a process for making a contract between two modules for the Order handling module in order to ask the service order to the Service configuration & activation module. When the contract between the Order handling module and the Service configuration & activation is agreed with, the service requester (the Order handling module) can send the service order request to the service provider (the Service configuration & activation module).

We define the sample scenario using WS-BPEL. First, we define the process entities in the partnerLinks, messages from senders and receivers in the variables, and management faults in the faultHandler. We also define the process sequence which is executed when the messages from some partnerLinks arrive in the sequence.

C. Contract

In this section, we define the contract in the XML format. The most important part in five parts of contract (General, Functional, Non-functional, Management and View specific) is the functional part. We need to define input, output variables and conditions in the functional part.

Figure 4 describes the system view contract of 'Request Customer Order' between the Order handling module and the Service configuration & activation module for the DSL fulfillment in the XML format. The General part contains contract names, versions and explanations and the Functional part contains Pre-condition, Post-condition, Input & Output Parameters and so on. The pre-condition checks that 'the customer is already registered' and 'the request is compatible with the customer's profile and feasible to network resources'. The Non-Functional and Management parts need to be defined when necessary.

```
<Contract>
  <General>
    ...
    <Descriptive>
      ...
      <Search_Criteria> Customer information, Associated_System_Processes </Search_Criteria>
    </Descriptive>
  </General>
  <View name='System_View'>
    <Functional>
      <Associated_System_Processes> manage & track customer, issue customer order </Associated_System_Processes>
      <Associated_System_Policies> </Associated_System_Policies>
      <System_Capabilities>
        <Input_Entities> CustomerOrderRequest </Input_Entities>
        <Output_Entities> CustomerOrderResponse </Output_Entities>
        <Pre-Conditions> customerIsRegistered, requestIsCompatible, requestIsFeasible <Pre-Conditions>
        <Termination> Successful </Termination>
        <Post-Conditions> Customer order is issued accordingly </Post-Conditions>
        <Post-Conditions_System_Exceptions> None identified </Post-Conditions_System_Exceptions>
        ...
      </System_Capabilities>
    </Functional>
    <Non-Functional> ... </Non-Functional>
    <Management> ... </Management>
  </View>
</Contract>
```

Figure 4. Contract: XML Format

D. Prototype Implementation

We have implemented NGOSS TNA components and management functions using Web Services technologies based on our proposed design. Our system has been implemented on a Linux server. We used Apache Web services project package [8], which is Java-based Web services software.

We first implemented the framework services using existing UDDI APIs such as save_business, save_service, find_business, find_service, get_serviceDetail, etc. We deployed these framework service functions into Web services using a Java Web Service provided by an apache SOAP engine. Then, we define WSDLs and implement management function modules such as order handling, service configuration, etc. Finally, we define the business process sequence in WS-BPEL and the BPEL engine executes the process in accordance with the process definition.

The execution step of our order handling for a DSL fulfillment is as follows: A customer orders a new DSL service through the service application form of Web-based user interface. This request message is handed over to the BPEL engine, and the engine checks the request message and the partnership, then, it directly calls the order handling service. The management processes are conducted in the sequence of Figure 3 and the BEPL engine processes these management sequences.

VI. CONCLUDING REMARKS

We have hexamined the concept, architectural principles and components of the NGOSS TNA. We proposed our technology-specific architecture using Web services technologies considering the requirement of NGOSS TNA. Web service is a distributed and services-oriented computing technology that can be applied to the NGOSS TNA. Also, we showed a case study based on our proposed design.

We will extend our prototype of Web services-based TSA and the management functions and also perform a test on our implementation system. Finally, we will extract performance metrics of Web services-based TSA and conduct performance analysis.

ACKNOWLEDGMENTS

This research was supported in part by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) (IITA-2005-C1090-0501-0018), by the Electrical and Computer Engineering Division at POSTECH under the BK21 program of the Ministry of Education, Korea, and by the KT Network Technology Lab.

REFERENCES

[1] TM Forum Technical Program, "NGOSS (New Generation Operations Systems and Software)", http://www.tmforum.org/

[2] TM Forum, "NGOSS Technology Neutral Architecture" - TMF053 v5.3 r6.0, Nov. 2005

[3] TM Forum, "NGOSS Phase 1 Technology Application Note-XML", TMF057 v1.5, Dec. 2001

[4] TMF, "NGOSS Phase 1 Technology Application Note - CORBA", TMF055, Version 1.5, Aug. 2001

[5] C. Ashford, "OSS through Java as an Implementation of NGOSS", White paper, Apr. 2004

[6] Mi-Jung Choi, Hong-Taek Ju, James W. K. Hong, and Dong-Sik Yun, "Towards Realization of Web Services-based TSA from NGOSS TNA", IPOM 2006, LNCS 4268, Dublin, Ireland, Oct., 2006, pp. 222-227.

[7] TM Forum, "Shared Information/Data (SID) Model", GB 922, Release 6.0, Nov. 2005

[8] Apache Software Foundation, "Web Services Project @ Apache", http://ws.apache.org, Refer Jan. 2007.