Proceedings

Workshop on
Service Oriented Architecture

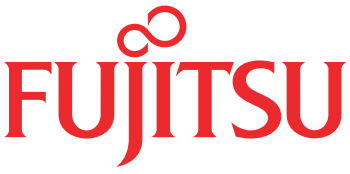
Sponsored by

Information Processing Society of Japan (IPSJ),
Special Interest Group on Software Engineering (SIG-SE)

In cooperation with
The Institute of Electronics, Information and Communication Engineers (IEICE)
Special Interest Group on Software Science (SIG-SS)
The Institute of Electronics, Information and Communication Engineers (IEICE)
Special Interest Group on Knowledge-based Software Engineering (SIG-KBSE)
Japan Society for Software Science and Technology (JSSST), FOSE


Microsoft


HITACHI
Inspire the Next


FUJITSU


Google


UNIVERSITAS NANZAN
N
ZAN
N
HOMINIS DIGNITATI

Preface

This volume contains the proceedings of the Workshop on Service Oriented Architecture, collocated with 14th Asia-Pacific Software Engineering Conference (APSEC 2007), which took place in Nagoya, Japan, December 4, 2007.

This workshop focuses on Service Oriented Architecture (SOA). Service oriented architecture is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations. By following service oriented architecture, a computer system or software is componentized as a service. One of the most important points of service oriented architecture is interoperability; the services should be accessed from other services easily. Service oriented architecture is still an emerging approach. There have not been so many concrete implementations yet. More researches and practices are required in this area.

We would like to thank the APSEC 2007 organization for give us the opportunity to organize the workshop, especially to the Conference Co-Chairs, Eiiti Hanyuda and Mikio Aoyama, and to the Workshop Chair, Osamu Shigo, for their assistance and continuous support. Many thanks to all those that submitted papers, and particularly to the contributing authors. Our gratitude also goes to the paper reviewers and the member of the Program Committee of the workshop, who helped in choosing and improving the selected papers. Information Processing Society of Japan (IPSJ) should be acknowledged for sponsoring the workshop.

December 2007

Workshop Co-Chair: William C. Chu
Toshiro Takase

Organizations

Workshop Co-Chairs:

William C. Chu	Tung Hai University, Taiwan
Toshiro Takase	IBM Tokyo Research Laboratory, Japan

Program Committee:

Aditya K. Ghose	University of Wollongong, Australia
Jun Han	Swinburne University of Technology, Australia
Yanbo Han	Institute of Computing Technology, Chinese Academy of Sciences, China
Fuyuki Ishikawa	National Institute of Informatics, Japan
Wei Jun	Institute of Software, Chinese Academy of Sciences, China
Piyush Maheshwari	IBM India Research Laboratory, India
Takahide Matsutsuka	Fujitsu Laboratories, Japan
Masahide Nakamura	Kobe University, Japan
Srinivas Padmanabhuni	Infosys, India
Stefan Tai	Karlsruhe University, Germany
Liang-Jie Zhang	IBM T. J. Watson Research Center, USA

Table of Contents

A Re-engineering a Legacy EDI System into SOA-based System: A Case Study*

T.C. Ho[1], C.W. Chu[1], C. Shih[1], N.L. Hsueh[2], C.H. Chang[3]
[1]*Department of Computer Science and Information Engineering , Tunghai University Taichung, Taiwan , ROC*
[2]*Department of Information Engineering and Computer Science, Feng Chia University Taichung, Taiwan, ROC*
[3]*Department of Information Management Hsiuping Institute of Technology,Taichung, Taiwan, ROC*
[1]*{g942917,cchu, shihc} @thu.edu.tw*
[2]*nlhsueh@fcu.edu.tw*
[3]*chchang@mail.hit.edu.tw*

Abstract

Nowadays the biggest challenge for the enterprise is how to integrate internal resource with business process effectively in such competitive market and to accomplish the advantage of competition. However, most systems usually couldn't satisfy the flexibility and efficiency of the business process. Service- Oriented Architecture(SOA) recomposes the individual function of the system, and help the enterprise reorganize fast and reuse highly to satisfy various business needs. In this paper, we will use Web service as the access interface and Unified Modeling Language(UML) for software molding to build multi-layer architecture of Electronic Data Interchange(EDI) system to fully integrate the internal Enterprise Resource Planning(ERP) system and EDI system for various business needs.

1. Introduction

1.1 Research Motive

In this high efficiency environment, the electronic transaction has brought intensely competition. The flexibility of the business process is the key to affect the enterprise success or not. In order to deal with various business needs, there are many heterogeneous applications in the enterprise. However, it is difficult to integrate the heterogeneous applications that develop with different languages. Therefore, the biggest challenge is how to integrate the business process and information technology for the enterprise. For this reason, we will use Service-Oriented Architecture and Web services to achieve system integration.

1.2 Research Purpose

Many companies use ERP systems to integrate daily operations such as finance, manufacturing, marketing and purchasing, etc. But the primary purpose of ERP system is to integrate the internal resource. The only way to gain more profits in competitive environment is to integrate supplier chain management and customer relationship management of the company. In traditional purchasing process, the buyer input the purchase data then print out the order to seller by email or fax. This traditional approach not only requires a great deal of human cost but also increase error rate.

Electronic Data Interchange (EDI) is the computer-to-computer exchange of business information in a standard format to enhance the operation efficiency. The traditional EDI system contains two major components: (1) EDI translation software that converts EDI format to internal applications, and (2) communication channels that deliver EDI documents to the trading partners. However, traditional EDI is too complicated and expensive for many companies.

* This research was sponsored by National Science Council, R.O.C. under the grant NSC 96-2218-E-035 -002

Nowadays company can be conducted in new ways like extensible markup language (XML). The primary purpose of XML is to facilitate the sharing of data across different information systems, particularly via the internet. The simplicity of XML also makes XML programmers easy to train and maintain.

Connecting each operation of internal resources is very important for the company especially when they make business decision. The purpose of this paper is how to design an EDI system that could compose fast and reuse highly to save human cost and improve data interchange. The enterprise could base on this architecture to expand the electronic commerce and supply chain system to enhance their competence with suppliers and customer.

2. Related Work

2.1 Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) is an architecture whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired results for a service consumer [8]. SOA defines how two computing entities, such as programs, interact in such a way as to enable one entity to perform a unit of work on behalf of another entity. Basic SOA model is defined by service providers and service consumers teamed with a service registry (figure 1)[8,9]. The service registry allows service consumers to find, access and invoke services that meet defined requirement. It also enables service providers to expose and advertise service offerings. The service description is written in an XML grammar called WSDL(Web Service Description Language) that defines the format of message.

In [11] Sneed presents a tool supported method for wrapping legacy code behind an XML shell by using service-oriented technology which allows individual functions within the programs, to be offered as Web services to any external users. In [4] Botta, Lazzerini, Marceloni present an approach to integrate a few service-oriented technologies such as Java Message Service and SAP Java Connector in order to exchange internal business data among several processes which use different communication protocols(HTTP, POP3, RMI, SMS). In [10] Sanchez-Nielsen, Martin-Ruiz and Rodriguez-Pedrianes propose a service-oriented architecture enlarge the variety of accessible services and new business opportunities in the mobile space that dynamical integration and discovery of available services at any time.

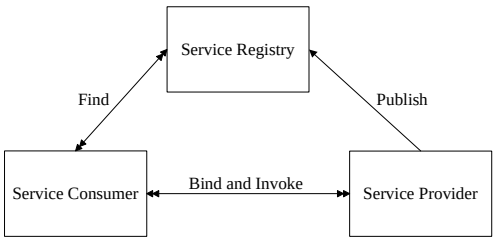


Figure 1. SOA operation diagram

2.2 Web services

As companies moving into collaborative e-commerce, they will have to inspect their own internal systems and applications. These applications must be able to communicate with the outside world. Web service is a software system designed to support interoperable machine-to-machine interaction over a network [1, 5]. Other systems interact with the Web services in a manner prescribed by its description using Simple Object Access Protocol (SOAP) messages. Web services offered within the framework of a service-oriented architecture promise to make applications more flexible, easy to compose and reuse. Web service is one of the ways you can implement SOA. The benefit of implementing SOA with Web services is that you achieve better interoperability with trading partners.

In [12] Yu propose a reusable access control layer for Web service software. This layer may be used for Web service software authorization. Access rights may easily be defined by a set of rules.

2.3 Unified Modeling Language (UML)

In the past we lack of common notation to show the analysis and design documents, so it is difficult to find the reuse from the source code. Unified Modeling Language (UML) is a standardized specification language for object modeling. It includes a graphical notation use to create an abstract model for a system [3]. UML improve communication between developer and architects and enhance system maintainability.

In [2] Arisholm, Briand, Hove and Labich have been investigated the impact of UML documentation on software maintenance. Experimental results show that, for complex tasks and past a certain learning curve, the availability of UML documentation may result in significant improvements in the functional correctness. In [7] Gomaa and Menasce have proposed an approach for the design and performance modeling of component interconnection patterns. These designs are performance annotated using an XML-type notation such as workload, system resources. They

then map the performance annotated UML design model to a performance model, which allows to analyze the performance of the software architecture executing on various system configurations.

2.4 Re-engineering

Re-engineering is a systematic way of examination and alteration of a subject system to reconstitute it in a new form and the subsequent implement of the new form[13].The objective of re-engineering is to re-structure or re-write legacy systems into new systems with better maintainability, flexibility and scalability [14].

Program transformations come in two forms: procedural (arbitrary functions applied to compiler data structures) and source-to-source (maps between concrete surface syntax forms of code). Procedural transformations have enjoyed enormous success for some 50 years in classical compilers [16] for translating source code to binary and optimizing code generators. Source-to-source program transformations were originally conceived as a method of program generation in the 1970s [17]. The seminal Draco system [18] provided the notion of domain analysis to support domain-specific language definition and transformational program generation. Draco accepted language (‘‘domain’’) definitions and true source to source transformations for each language, using recursive-descent parsing. This type of technology has continued development, moving into scientific computing with the Sinapse system [19], which has evolved into a commercial system for synthesizing sophisticated financial instruments [20].

In [21] R.L. Akers has constructed highly scalable transformation tools supporting modern languages in full generality. They can leverage the infrastructure costs by developing a common transformation system infrastructure re-useable by derived tools that each address specific tasks.

3. Research Methods

3.1 Re-engineering Process

The process can be divided into two processes: reverse engineering process and forward engineering process. The reverse engineering process as discuss in [15]. The process can be divided into four phases: Context Parsing, Component Analyzing, Design Recovering and Design Reconstructing. According this process, we can recover the design specification of components.

After these processes, we can repack the component as SOA component using forward engineering as discussed in section 3.3.

3.2 System Architecture

The infrastructure of SOA is to create service component. We propose the framework that could encapsulate business process and output the result with message format to the caller. We also describe the service purpose, input message and output message of the service component to show the instruction for service consumer.

SOA is a distributed architecture, so we design the framework based on four layers. The presentation layer provides a communication bridge between end user and system. It creates the graphical views of the objects, and captures user inputs. We could design various user interfaces such as mobile, PDA, and home appliances. For this reason, we separate the presentation layer from the system architecture. It could develop various user interfaces and reduce the impact of system change and business logic change. The service layer is to compose the required components to generate the Web service according to the needs of the enterprise. In the component layer, we will encapsulate business logic inside reusable classes or components. Each component has the specific responsibility to provide the specific function. Furthermore, we also apply Microsoft Message Queue (MSMQ) technology to store messages in MQ Server. Applications send messages to MQ Server, and MQ Server stores the messages in queues for later delivery. This way not only ensures the information security via encryption technology but also receives asynchronous data. The responsibility of the data access layer is to provide a simple and safe interface by encapsulating complex communication process between the component layer and database storage. This way also can reduce the impact of the database storage change. The focus of this paper is to design the standard components in the component layer and compose the services in the service layer.

3.3 Research Steps

In this section, we will construct an Electronic Data Interchange system with high extendibility, easy maintenance, and better reusability to integrate the ERP system of the enterprise based on Service-Oriented Architecture. Our research steps are as following:

(1) Step 1: Process Analysis- to draw the use case

diagrams.

(2) Step 2: System Design- to design the standard components and draw the class diagram and sequence diagrams.

(3) Step 3: Service Composition- to create simple services and business services with the standard components.

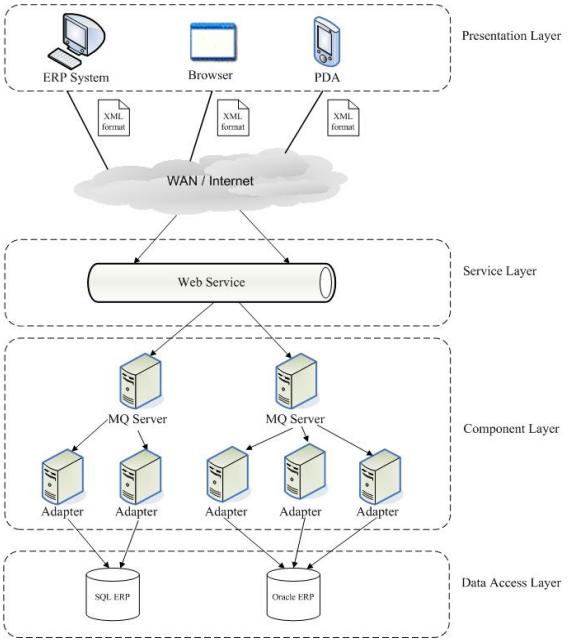


Figure 2. Architecture diagram of EDI system

3.3.1. Step 1 Process Analysis. According the reverse engineering processes which we discuss in section 3.1, we can recover the design specification of legacy system. In this case study, we focus on the re-engineering the customer purchasing system. First, we analyze the use case of sending purchase order, receiving purchase acceptance, sending payment payment. Hence, we draw the use case diagram according to the three use cases (Figure 3). Second, we analyze the business process of each scenario of these use cases. For example the use case of receiving purchase acceptance, we write use case description to show the behavior between actors and system according to each scenario.

(1) The purchasing agent received the data of purchase acceptance.

(2) The system searched the price of the purchase order.

(3) The system calculated the sum of money and taxes.

(4) The system updated the quantity of the purchase order according to the purchase number.

(5) The system updated the quantity of the inventory according to the product number.

(6) The system generated the new number by adding one number to maximum number.

(7) The system inserted data of purchase acceptance.

(8) The system inserted data of purchase acceptance detail.

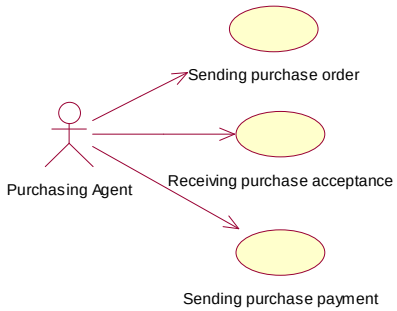


Figure 3. Use case diagram

3.3.2. Step 2 System Design. In this step, we find out the static elements of design phase such as classes, attributes, and relationships between classes according to the use case description. Hence, we draw a class diagram (figure 4) that describes the structure of the system by showing the system's classes, their attributes, and the relationships between classes. The class description is shown in table 1.

3.3.3. Step 3 Service Composition. In order to provide simple interface for clients, we use façade pattern [6] to encapsulate complex processes of the components in design phase. We divide the service type into simple service and business service, therefore to reuse component and reduce complexity of the system. Simple service refers to basic service (ex. query purchase order) and nontransaction service (ex. calculate the sum of money and taxes). Business service refers to combine multiple simple services or multiple components to become complex business process (ex. receive purchase acceptance). Business service is usually to correspond business process of the enterprise directly, hence we can customize any requirement of customers.

For example the use case of receiving purchase acceptance, we draw sequence diagram to illustrate events sequentially input from an external source according to use case description and Table 1. We, then, publish the interface by Web services.

(1) Simple services

The processes of simple services of this case contain the operations of getting purchase order and calculating the money and taxes of purchase order. We can compose the required components to become Web services such as IGetSinglePurchaseOrder, IGetSinglePurchaseOrderDetail, and

ICalculateAmount (figure 5).

Table 1. Class description

class	method	description
PurchaseOrder	GetData	to get data of purchase order
PurchaseOrderDetail	GetData UpdateDeliveryQty	to get data of purchase order detail to update quantity of purchase order detail
Product	GetData	to get data of product
Supplier	GetData	to get data of supplier
SupplierPrice	GetData GetDataBySupplier	to get data of supplier price to get data by supplier
PurchaseAcceptance	GetData InsertData GetMaxNo	to get data of purchase acceptance to insert data of purchase acceptance to get max number of purchase acceptance
PurchaseAcceptanceDetail	GetData InsertData	to get data of purchase acceptance detail to insert data of purchase acceptance detail
Inventory	GetData UpdateData	to get data of inventory to update data of inventory
PurchaseProcess	CalculateAmount GenerateNo	to calculate amount to generate the order number

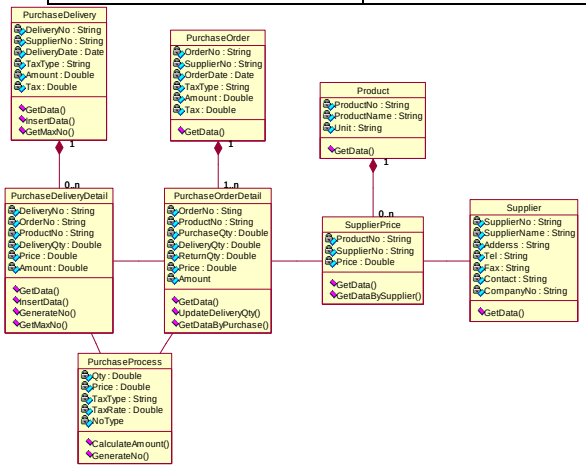


Figure 4. Class diagram

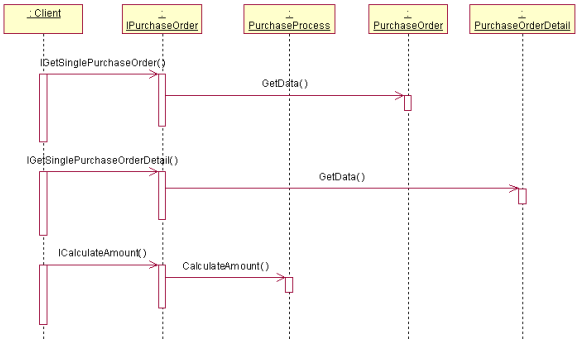


Figure 5. Sequence diagram of simple services

(2) Business services

The processes of business services of this case contain the operations of updating purchase order, updating quantity of inventory, inserting purchase acceptance, inserting purchase acceptance detail, and receiving purchase acceptance. We can compose the required components for the operations of updating purchase order, updating the quantity of inventory, inserting purchase acceptance, inserting purchase acceptance detail, and provide four interfaces to the use case of receiving purchase acceptance. The advantage of this way is service reusability, but it's difficult to maintain the multiply interfaces. Besides, we can compose the required components of the interface of IEDIInsertPurchaseAcceptance directly (figure 6). This way could not only encapsulate complex process but also easily simplify the access interface.

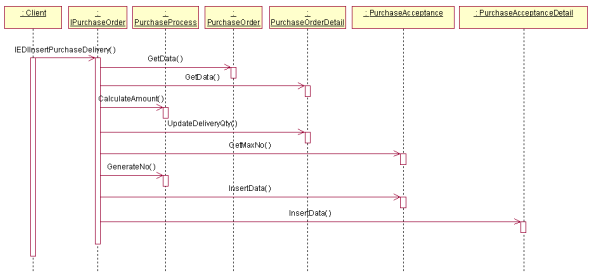


Figure 6. Sequence diagram of business services

4. Research Results

In order to demonstrate the improvement of this paper on software development, we coordinate with TenInfo System Ino. The company uses Microsoft Visual Studio .NET and the UML documentation to facilitate system development. Their primary service develops and customizes the ERP system, also assists the customer to integrate ERP system, Customer Relationship Management system (CRM), Supply

Chain Management system(SCM) and EDI system. The EDI system team contains two system analysts and three programmers to be responsible for the system development. Their original approach is to write specification through system analysts interview with customers and coding immediately. This way may save time in simple system, but will encounter difficulty on software maintenance. Therefore we compare our approach and the original approach in TenInfo System Ino. The results are summarized in table 2.

(1) Time: The original approach must establish additional transition system and communication system, hence it spends 50 days to complete the whole EDI system. The transition system spend about 15 days, the system development spends about 30 days, and the system testing spends about 5 days. Our approach spend 40 days to complete the while EDI system. Although the UML documentation spends about 15 days, the system development just 15 days, and the system testing about 10 days. The overall efficiency improves 25%.

(2) Functional correctness: The original approach is the programmers to code according to the simple analysis document directly. System engineers only used black box testing to validate the expected outputs, but not any additional test plan is written. The functional correctness is about 80%. Our approach uses UML documentation to facilitate the system development, and therefore system engineers can use test plan to validate the function correctness. However, we improve the whole system correctness.

(3) The participant's comment: We interview with two system analysts and three programmers. Because the original approach must establish the transition system and the communication system, so the cost to be quite high. There is no additional documentation to facilitate system development, therefore the system is difficult to debug. The different trading partners will be using different versions of EDI Directories, it is difficult to extend your business scope.

From the research result of this paper, the advantages of EDI system based on SOA are as followings:

(1) High maintainability: In system design phase, we use UML documentation to improve system maintainability and reduce maintenance cost.

(2) Low complexity: Using object-oriented technology is not only to simplify complex system but also achieve high reuse.

(3) High flexibility: The application based on SOA didn't need to consider that customer use what kind of program language, so the acceptability is higher. We could customize any needs by designing standard components and achieve high flexibility on related system development.

Table 2. Summary of Research Results

Topics	Original approach	SOA approach
Time	50 days	40 days
Function correctness	80%	95%
The participant's comment	Low acceptance Difficult to maintain Difficult to debug Special message format Difficult to expand	High acceptance Easy to maintain Easy to debug Easy to customize

5. Conclusion

When the enterprise faces internationalization environment the biggest challenge is to integrate internal and external resources to create more benefits, therefore Electronic Commerce becomes a necessary part. Electronic Data Interchange is the core of business-to-business, it is not only to integrate business-to-business process efficiently via computer and internet but also save unnecessary human cost and manual error to improve managing benefit. In this paper, we integrate all resources quickly to facilitate system development via Service-Oriented Architecture.

6. References

[1] G. Alonso, F. Casati , H. Kuno, and V. Machiraju, ***Web Services Concepts, Architectures and Applications***, Springer Verlag. 2004.

[2] E. Arisholm, L.C. Briand, S.E. Hove, Y. Labiche, "The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation," ***IEEE Transactions on Software Engineering***, Vol. 32, No. 6, pp.365-381, June 2006

[3] G. Booch, J. Rumbaugh, and I. Jacobson, ***The Unified Modeling Language User Guide***, Addison-Wesley, Reading, 1999.

[4] A. Botta, B. Lazzerini, and F. Marcelloni, "Integrating service-oriented technologies to support business processes," ***Proceeding of the 2005 Seventh IEEE International Conference on E-Commerce Technology Workshops***(CECW'05), 2005.

[5] M. Champion , C. Ferris, E. Newcomer, and D. Orchard, "Web services architecture", http://www.w3.org/TR/ws-arch/

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, ***Design Patterns: Elements of Reusable Object-Oriented Software***, Addison-Wesley, 1995.

[7] H. Gomaa and D.A. Menasce, "Design and Performance Modeling of Component Interconnection Patterns for Distributed Software Architecture," *Proc. Second Int'l Workshop Software and Performance (WOSP2000)*, pp. 117-126, Sept. 2000

[8] S. Hashimi, *Service-Oriented Architecture Explained*, O'Reilly ONDotnet.com, August 2003, http://www.ondotnet.com/pub/a/dotnet/2003/08/18/soa_explained.html

[9] M. Luo, M. Endrei, P. Comte, P. Krogdahl, J. Ang, and T. Newling, *Patterns: Service-Oriented Architecture and Web Services*, April. 2004, http://www.redbooks.ibm.com/abstracts/sg246303.html?Open

[10] E. Sanchez-Nielsen, S. Martin-Ruiz, and J. Rodriguez-Pedrianes, "An Open and Dynamical Service Oriented Architecture for Supporting Mobile Services," ACM 1-59593-352-2/06/0007(ICWE'06)

[11] H.M. Sneed, "Integrating legacy Software into a Service oriented Architecture," *Proceedings of the Conference on Software Maintenance and Reengineering(CSMR'06)*, 2006.

[12] W.D. Yu, "An Intelligent Access Control for Web Services Based on Service Oriented Architecture Platform," *Proceedings of the Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems and Second International Workshop on Collaborative Computing, Integration, and Assurance* (SEUS-WCCIA'06), 2006.

[13] E.J. Chikofsky, and J. H. Cross, "Reverse Engineering and design Recovery: A Taxonomy," *IEEE Software*, Vol. 7 No. 1, pp. 13-17, 1990.

[14] I. Sommerville, *Software Engineering*, Fifth Edition, Addison-Wesley Publishing Co. Inc., Wokingham, England, pp. 660-663, 700-703, 1996.

[15] C.W. Lu, C.W. Chu, C.H. Chang, Y.C. Chung, X. Liu, and H. Yang, "Reverse engineering," *Handbook on Software Engineering and Knowledge Engineering Vol. II.* World Scientific Publishing: River Edge, NJ, 447 – 466, 2002.

[16] A. Aho, R. Sethi, J. Ullman, Compilers: *Principles, Techniques and Tools*, Addison-Wesley, Reading, MA, 1986.

[17] R.M. Balzer, N.M. Goldman, D.S. Wile, "On the transformational implementation approach to programming, " in *Proceedings, 2nd International Conference on Software Engineering*, Oct. 1976, pp. 337–344.

[18] J. Neighbors, "Draco: a method for engineering reusable software systems," *Software Reusability*, T. Biggerstaff, A. Perlis (Eds.), ACM Press, 1989.

[19] E. Kant, F. Daube, E. MacGregor, J. Wald, "Scientific programming by automated synthesis," *Automating Software Design*, Michael R. Lowery, Robert D. McCartney (Eds.), MIT Press, 1991.

[20] www.scicomp.com, SciFinance pricing model development environment.

[21] L. Robert. A. Akers, I. D. Baxter, M. Mehlich, B.J. Ellis, K.R. Luecke, "Case study: Re-engineering C++ component models via automatic program transformation", *Information and Software Technology*, 49 pp.275 – 291, 2007.

Communication Centered Programming of Integrated Services with Priority in Home Appliance Network

Sakura Bhandari† Shoji Yuen†† Kiyoshi Agusa†††
Graduate School of Information Science, Nagoya University, Japan
Furo-cho,Chikusa-ku, Nagoya, Japan
†sakura@agusa.i.is.nagoya-u.ac.jp, {††yuen,†††agusa}@is.nagoya-u.ac.jp

Abstract

We propose an extended framework of communication centered programming for home appliance network where the distributed environment is designed in the service oriented architecture. Home appliances are now getting connected to network. By exchanging messages over network, various appliances constitute "integrated services" in that the operations by the appliances are performed as programmed. Following the communication centered programming[5, 6], a global description is directly given for interactions that involve many appliances. A behaviorally equivalent local description distributed for the appliances is derived by the end-point projection(EPP). This significantly eases the distributed system programming. The communication centered programming is originally proposed by Carbone et.al. Services are available at the same time and a certain service may or must have the precedence over the other services to be consistent. To cope with this, we introduce "priority" in the communication centered programming. We present a design extension for priority both in global descriptions and local descriptions by examples. We argue the end-point projection with priority towards the automatic derivation of local descriptions from global descriptions.

1. Introduction

Recently, a variety of electric home appliances, such as DVD, Video, refrigerator, or even toaster, have become connected to network. Homes are being equipped with such networked appliances to allow a more convenient way of living by offering services that integrate various operations. As appliances are getting more sophisticated and diversified, it is suggested that each appliance works on the SOA basis. Due to the dynamic nature of home appliance network, the interactions are conducted by the appliances without any centralized server, where the appliances can be dynamically added and modified. All features provided by the appliances are encapsulated a self-contained object, which is *loosely coupled* with other objects. Inherently, it is not easy to realize the precise state of all the appliances over network. Precisely, controlling the appliances may lead to the difficulty of definition and maintenance in the distributed environment. We focus on the framework called "communication centered programming" proposed by Carbone et.al[5, 6]. The framework works as follows: first, the overall behavior across the network is given as a global description, originated from Choreography Description Language(CDL), developed by W3C's WS-CDL Working Group. It is then translated into local descriptions that define the operations of individual appliances. Assuming that a global description is well-structured, it is known that there is a systematic translation from a global description to the corresponding local description. This translation is called the *end point projection, EPP* for short.

As for integrated services of home appliances, the idea of communication centered programming fits the purpose of its definition. However, when couple of interactions are performed at the same time, conflicts among the interactions may occur. We call such conflicts *interference problem* of interactions in integrated services. Usually, such interference has to be resolved depending on the use of the appliance by users or the environment. To resolve interference problem, we add *priority* for interactions. When an interference exists among the interactions, according to the demand of the users, a priority value is assigned to each component in the global description. The global description with priority can be translated to the local description with priority, with the help of EPP. The projection from global descriptions to local descriptions should preserve the behavior with priority in the consistent manner. Here, we do not assume any global priority resolving mechanism. The priority is only *locally* resolved. As syntax, we extend global descriptions by adding the "priority altering operator" and for local descriptions we add "priority guard" originally proposed by Phillips[2].

The paper is structured as follows. In section 2, we review the communication centered programming in home

appliance network. Section 3 argues the interference problem and section 4 presents the basic idea of priority. In section 5, we present the extension of both global and local descriptions. We argue the EPP with priority in section 6 and in section 7, we give the concluding remarks and future work.

2. Communication Centered Programming in Home Appliance Network

SOA is a system architecture to integrate different systems distributed over a network. Each system exports own features to the network as a unit of service(a set of task which is coarser than an object). In this system, we can assume that each networked appliance exports its own services to the network. The users are allowed to create integrated services using the services the appliances export. The communication centered programming[5] suggests the two layered descriptions: *global description* and *local description*. In our setting, a global description defines a flow of communications between appliances, users and the environment. We call this flow of communications *interaction*. A local description is a composition of the participant's behavior. The behavior of an appliance directly defines the local operation. The behavior of a user defines the user operation, and the behavior of the environment defines the possible change of environment, such as the change of daylight.

We now show the overview of the system we consider for the home appliance network. A global description describes an interaction scenario from a vantage viewpoint. With the help of EPP, such global descriptions are transformed to local descriptions of the network appliances. The local descriptions precisely identify a local behaviour of each appliance. [5] explores a theory of EPP which defines three principles for well-structured global description. It has been confirmed that the integrated services can be written with the help of such descriptions, and the global descriptions are also well-structured[9].

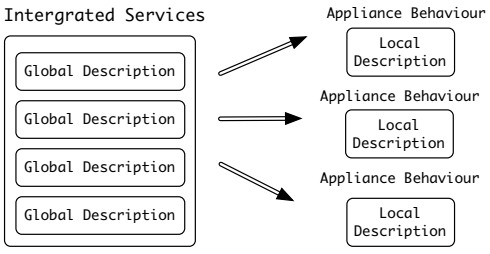


Figure 1. Network Appliance System

3. Interference between Integrated Services

We first show a couple of examples of some typical interactions with interference.

- **DVD Play Scenario(IS_1):** When a user plays the DVD, the TV sets to the video mode, the light is turned off, and the curtains are closed.
- **Coming Home Scenario(IS_2):** When a user comes home, the air-conditioner and the light turn on, and the recorded message on the phone starts to play.

Imagine the following typical scenario at home. User1 wants to watch a DVD, he switches on the DVD player and in a short time, IS_1 starts to operate. The DVD starts to play, the TV is set to the video mode, the light turns off and the curtains are closed. Now, assume that another user, user2, comes back home. This time interaction scenario IS_2 starts to operate. One should notice that user1 is still watching the DVD and the appliances are working according to IS_1. When IS_2 starts to operate, the air-conditioner and the light turns on, and the phone's recorded message starts to play. Here, the light which was *turned off* by IS_1 is *turned on* by IS_2, which can be unexpected for user1. We can therefore examine the interference between the interactions. In a network, appliances may be controlled by more than one service, and indeed these controlling services are often trying to achieve different goals. Therefore, we define *interference* as a phenomenon, where two interactions include two different functions of a common appliance. Figure2 shows an example of an interference detection between the DVD Play Scenario(IS_1) and the Coming Home Scenario(IS_2), where IS_1 and IS_2 have the *small* and the *on* functions respectively, of a common appliance LIGHT1. To cope with the unexpected interferences, there is a need to detect and resolve such interferences beforehand. Although such interference should be resolved, currently there is no way to resolve it in CDL. Adding the priority mechanism, the operational semantics of CDL is enhanced where more control can be specified by developers for interfering operations. During the runtime, the interaction with a higher priority becomes ready. Operations with same priority are executed nondeterminstically.

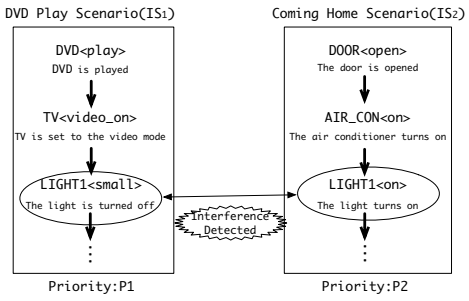


Figure 2. Detection of Interference between Two Interactions

The interference shown in Figure2 is resolved by adding the higher priority to IS_1 intending to continue watching

DVD.

4. Design for Integrated Services

In our system, we have shown an example of 8 interactions in Figure3(I1, I2,..., I8). One may notice that some of the interactions are in a *relative relation*, that is, a beginning of an interaction can be considered as the termination of another interaction. Hence, such interactions cannot occur simultaneously. For example, when a DVD is stopped(I2), it means that the interaction for playing the DVD(I1) is not longer *alive*; we can assume that the DVD is no longer played. Similarly, coming home interaction(I3) is in a relative relation with going out interaction(I4); when a user comes back, it is assumed that the going out service was finished(the user has come back home). Therefore, in our system, such relative interactions are treated as one *service*. For example, the interaction of playing the DVD(I1) and stopping the DVD(I2) are treated as one service, **DVD service(S1)**. Similarly, two interactions I3 and I4 in Figure3 can be handled as **IN-OUT service(S2)**. With 8 examples of interactions and observing the relative relations between these examples, we could achieve 5 services(S1-S5) as shown in Figure3.

- **S1:DVD Service**:

  I1:When a user plays the DVD, the TV sets the video mode, the light is turned off, and the curtains are closed.

  I2:When a user stops the DVD, the TV is set to the channel mode, the light is turned on, and the curtains are opened.

- **S2:IN-OUT Service**:

  I3:When a user comes home, the air-conditioner and the light turns on, and the recorded message on the phone starts to play.

  I4:When a users goes out of the home, the DVD player, TV, air-conditioner and the light are turned off and the curtains are shutdown.

- **S3:DAY Service...**
- **S4:TELEPHONE Service...**
- **S5:RECORD Service...**

Figure 3. Scenario of Integrated Services

After detecting the interferences between 8 interactions, we may assign the priority value to each interaction as follows, before the interactions are actually ready to execute.

Service	S1		S2		S3		S4	S5
Interaction	I1	I2	I3	I4	I5	I6	I7	I8
Priority	p1	p3	p2	p3	p4	p2	p3	p2

where $\texttt{p1} > \texttt{p2} > \texttt{p3} > \texttt{p4}$. Here, $p_i > p_j$ means p_i has the higher priority than p_j. A priority is assigned to each interaction, which belongs to a certain service. As a service may contain more than one interaction, the priority of a service is equal to the priority of its interaction that is executing at that moment. Therefore, the priority of a service can be overwritten by its current executing interaction. This priority assignment resolves the interference as shown in Figure2 as follows. Figure2 shows the interference between DVD play interaction(I1) and coming home interaction(I3) as we described in Section3. As shown in Figure2, I1 has a priority value p1, and I3 has a priority value p2. When a user comes back home during another user is still watching a DVD, the light which was turned off by I1 will not turn on by the operation of I3 because the priority value of I3(p2) is smaller than that of I1(p1). Interactions like I1 and I6 may have a same priority value, since they are independent in their behavior .

5. Descriptions with Priority

5.1. Global Description with Priority

The syntax of the global description is given by the BNF in Figure4. *IS* denotes the class of integrated services. A service is an interaction I labelled by N written as $N : I$. For example, a DVD service consists of a pair of alternative interactions: one to start playing the DVD and the other to stop playing the DVD. Each interaction may involve two or more participants, shown as A, B in (3). A channel is ranged over by ch or ch', and a session by s or t. Sessions are initiated through the channels particular to the participants, and the participants communicate in each session. (3) is the initiation of a session under a channel, and (4) is a communication between A and B through session s. (7) is the *priority altering operator*. Here, *pmap* maps the priority to all its arguments. For the binary case, when pmap(1) = p1, pmap(2) = p2 , we write $P_1\ {}_{p_1+p_2} P_2$. $\mathcal{PA}$ denotes the set of the participants and $\mathcal{SN}$ denotes the set of the services. $\mathcal{PN}$ indicates the set of priority values. $\mathcal{PN}$ is the set of integers with the special value init which is the biggest value for initiation of session.

The operational semantics of global descriptions is defined by extending that of [5] with the priority configuration. The reduction relation is defined by the rules in Figure5. The reduction is in the following form:

$$\langle \texttt{I}, \sigma, \rho, \pi \rangle \rightarrow \langle \texttt{I}', \sigma', \rho', \pi' \rangle$$

which says that I in the state σ, with functions ρ, π performs one-step computation and becomes I' and the new state σ' with the new priority configuration ρ' and π'. σ and σ' : $\mathcal{X} \rightarrow \mathcal{V}$ are the states of variables of participants. The ρ and $\rho' : \mathcal{PA} \rightarrow \mathcal{PR} \times \mathcal{SN}$ assigns the *priority value* and the *service's name* to which the priority value belongs A service can only use the appliance if the service's priority value is higher than that of the appliance, or the appliance is engaged by the service. The π and $\pi' : \mathcal{SN} \rightarrow \mathcal{PN}$ which allocate priority values to services.

$$\frac{\mathtt{e@A} \Downarrow \mathtt{v}, \quad \pi(\mathtt{S}) = \mathtt{Init}}{\langle \mathtt{S} : \mathtt{A} \rightarrow \mathtt{B} : \mathtt{s}\langle \mathtt{op}, \mathtt{e}, \mathtt{x}\rangle.\mathtt{I},\ \sigma,\ \rho,\ \pi\rangle \quad \rightarrow \quad \langle \mathtt{S} : \mathtt{I},\ \sigma[\mathtt{v}/\mathtt{x}],\ \rho,\ \pi\rangle}(\mathtt{COMM1}_{\mathtt{init}})$$

$$\frac{\pi(\mathtt{S}) = \mathtt{Init}}{\langle \mathtt{S} : \mathtt{A} \rightarrow \mathtt{B} : \mathtt{ch}(\nu \mathtt{s})).\mathtt{I},\ \sigma,\ \rho,\ \pi\rangle \quad \rightarrow \quad \langle \mathtt{S} : (\nu \mathtt{s})\mathtt{I},\ \sigma,\ \rho,\ \pi\rangle}(\mathtt{COMM1}_{\mathtt{first}})$$

$$\frac{\mathtt{e@A} \Downarrow \mathtt{v}, \quad (\mathtt{P_A}, \mathtt{S_A}) = \rho(\mathtt{A}), \quad (\mathtt{P_B}, \mathtt{S_A}) = \rho(\mathtt{B}), \quad \mathtt{p_A} \leq \pi(\mathtt{S}) \vee \mathtt{S_A} = \mathtt{S}, \quad \mathtt{p_B} \leq \pi(\mathtt{S}) \vee \mathtt{S_B} = \mathtt{S}}{\langle \mathtt{S} : \mathtt{A} \rightarrow \mathtt{B} : \mathtt{s}\langle \mathtt{op}, \mathtt{e}, \mathtt{x}\rangle.\mathtt{I},\ \sigma,\ \rho,\ \pi\rangle \quad \rightarrow \quad \langle \mathtt{S} : \mathtt{I},\ \sigma[\mathtt{v}/\mathtt{x}], \rho[(\mathtt{p_1}, \mathtt{S})/\mathtt{A}, (\mathtt{p_1}, \mathtt{S})/\mathtt{B}],\ \pi\rangle}(\mathtt{COMM2})$$

$$\frac{\langle \mathtt{S_1} : \mathtt{I_1}, \sigma, \rho, \pi\rangle \rightarrow \langle \mathtt{S_1} : \mathtt{I_1'}, \sigma', \rho', \pi'\rangle, \quad \pi(\mathtt{S_1}) \geq \pi(\mathtt{S_2})}{\langle \mathtt{S_1} : \mathtt{I_1} \ \| \ \mathtt{S_2} : \mathtt{I_2}, \sigma, \rho, \pi\rangle \quad \rightarrow \quad \langle \mathtt{S_1} : \mathtt{I_1'} \ \| \ \mathtt{S_2} : \mathtt{I_2}, \sigma', \rho', \pi'\rangle}(\mathtt{PAR-L})$$

$$\frac{\langle \mathtt{S} : \mathtt{I}, \sigma, \rho, \pi\rangle \quad \rightarrow \quad \langle \mathtt{S} : \mathtt{I'}, \sigma', \rho', \pi'\rangle}{\langle \mathtt{S} : (\nu \mathtt{s})\mathtt{I}, \sigma, \rho, \pi\rangle \quad \rightarrow \quad \langle \mathtt{S} : (\nu \mathtt{s})\mathtt{I'}, \sigma', \rho', \pi'\rangle}(\mathtt{RES}) \qquad \frac{\mathtt{I}[\mathtt{recX.I}/\mathtt{X}] \quad \rightarrow \quad \mathtt{I'}}{\mathtt{recX.I} \quad \rightarrow \quad \mathtt{I'}}(\mathtt{REC})$$

$$\frac{\mathtt{e@A} \Downarrow \mathtt{tt}}{\langle \mathtt{S} : \mathtt{if\ e@A\ then\ I_1\ else\ I_2}, \sigma, \rho, \pi\rangle \quad \rightarrow \quad \langle \mathtt{S} : \mathtt{I_1}, \sigma, \rho, \pi\rangle}(\mathtt{Cond}_{\mathtt{true}})$$

$$\frac{\mathtt{e@A} \Downarrow \mathtt{ff}}{\langle \mathtt{S} : \mathtt{if\ e@A\ then\ I_1\ else\ I_2}, \sigma, \rho, \pi\rangle \quad \rightarrow \quad \langle \mathtt{S} : \mathtt{I_2}, \sigma, \rho, \pi\rangle}(\mathtt{Cond}_{\mathtt{false}})$$

$$\frac{\langle \mathtt{S} : \mathtt{I_i}, \sigma, \rho, \pi\rangle \quad \rightarrow \quad \langle \mathtt{S} : \mathtt{I_i'}, \sigma', \rho', \pi'\rangle}{\langle \Sigma_{\mathtt{pmap}} \mathtt{S_i} : \mathtt{I_i}, \sigma, \rho, \pi\rangle \quad \rightarrow \quad \langle \mathtt{S} : \mathtt{I_1'}, \sigma', \rho', \pi'[\mathtt{pmap}(\mathtt{p_i})/\mathtt{S}]\rangle}(\mathtt{ALT}_{\mathtt{Pri}})$$

Figure 5. Reduction Semantics for Global Description

$$\begin{aligned}
\mathtt{IS} &::= \mathtt{S_1} \parallel ... \parallel \mathtt{S_n} && (1)\\
\mathtt{S} &::= \mathtt{N} : \mathtt{I} && (2)\\
\mathtt{I} &::= \mathtt{A} \rightarrow \mathtt{B} : \mathtt{ch}(\nu \mathtt{s}).\mathtt{I} && (3)\\
&\ \ | \ \ \mathtt{A} \rightarrow \mathtt{B} : \mathtt{s}\langle \mathtt{op}, \mathtt{e}, \mathtt{x}\rangle.\mathtt{I} && (4)\\
&\ \ | \ \ \mathtt{x@A} := \mathtt{e.I} && (5)\\
&\ \ | \ \ \mathtt{if\ e@A\ then\ I_1\ else\ I_2} && (6)\\
&\ \ | \ \ \Sigma_{\mathtt{pmap}} \mathtt{I_i} && (7)\\
&\ \ | \ \ (\nu \mathtt{s})\mathtt{I} && (8)\\
&\ \ | \ \ \mathtt{X^A} && (9)\\
&\ \ | \ \ \mathtt{recX^A.I} && (10)
\end{aligned}$$

Figure 4. Syntax of the Global Description

Example

Figure6 is an example of a global description with priority for DVD service(S1) with the annotation for threads[5] [1].

This description consists of two parts; (11) to (12) where the session is initiated in the channel between the appliances, and lines from (13) through (16) show the actual interaction between the appliances. DVD service consists of two alternative interactions; one where the DVD is played, lines from (13) through (14), and the other where the DVD is stopped, lines from (15) through (16). The priority value for the former description is p1 and p3 for the later description. Therefore , the priority of such relatively related descriptions are expressed as $_{p1+p3}$ with the use of the priority altering operator. Here, $\pi(\mathtt{DVD_{SERVICE}}) = \mathtt{p1}$ when the DVD is played, and $\pi(\mathtt{DVD_{SERVICE}}) = \mathtt{p3}$ when the DVD is stopped. Moreover, when a DVD is played, $\rho(\mathtt{LIGHT1}) = (\mathtt{p1}, \mathtt{DVD_{SERVICE}})$ which shows that LIGHT1 has priority value of p1, set by $\mathtt{DVD_{SERVICE}}$. Similarly, when a DVD is stopped, $\rho(\mathtt{LIGHT1}) = (\mathtt{p3}, \mathtt{DVD_{SERVICE}})$ which indicates that LIGHT1 has a priority value of p3, set by $\mathtt{DVD_{SERVICE}}$. The priority of DVD service is overwritten according to which of its interaction is operating.

[1] The annotation is important to distribute the operations by EPP. Although we do not present the project here, the annotation shows the description is well structured in terms of EPP.

$\mathtt{DVD}_{\mathtt{SERVICE}}$: (

$\mathtt{USER}^1 \rightarrow \mathtt{DVD}^2 : \mathtt{ch}_{\mathtt{DVD}}(\nu\, \mathtt{s}_{\mathtt{DVD}})$. (11)

$\mathtt{DVD}^2 \rightarrow \mathtt{TV}^3 : \mathtt{ch}_{\mathtt{TV}}(\nu\, \mathtt{t}_{\mathtt{DVD}})$.

$\mathtt{TV}^3 \rightarrow \mathtt{LIGHT1}^4 : \mathtt{ch}_{\mathtt{LIGHT1}}(\nu\, \mathtt{u}_{\mathtt{DVD}})$.

$\mathtt{LIGHT1}^4 \rightarrow \mathtt{CURTAIN1}^5 : \mathtt{ch}_{\mathtt{CURTAIN1}}(\nu\, \mathtt{v}_{\mathtt{DVD}})$.

$\mathtt{CURTAIN1}^5 \rightarrow \mathtt{LIGHT1}^4 : \mathtt{v}_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$. $\mathtt{LIGHT1}^4 \rightarrow \mathtt{TV}^3 : \mathtt{u}_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$.

$\mathtt{TV}^3 \rightarrow \mathtt{DVD}^2 : \mathtt{t}_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$. $\mathtt{DVD}^2 \rightarrow \mathtt{USER}^1 : \mathtt{s}_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$.

$\mathtt{USER}^1 \rightarrow \mathtt{DVD}^6 : \mathtt{ch}_{\mathtt{DVD}}(\nu\, \mathtt{s}'_{\mathtt{DVD}})$.

$\mathtt{DVD}^6 \rightarrow \mathtt{TV}^7 : \mathtt{ch}_{\mathtt{TV}}(\nu\, \mathtt{t}'_{\mathtt{DVD}})$.

$\mathtt{TV}^7 \rightarrow \mathtt{LIGHT1}^8 : \mathtt{ch}_{\mathtt{LIGHT1}}(\nu\, \mathtt{u}'_{\mathtt{DVD}})$.

$\mathtt{LIGHT1}^8 \rightarrow \mathtt{CURTAIN1}^9 : \mathtt{ch}_{\mathtt{CURTAIN1}}(\nu\, \mathtt{v}'_{\mathtt{DVD}})$.

$\mathtt{CURTAIN1}^9 \rightarrow \mathtt{LIGHT1}^8 : \mathtt{v}'_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$. $\mathtt{LIGHT1}^8 \rightarrow \mathtt{TV}^7 : \mathtt{u}'_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$.

$\mathtt{TV}^7 \rightarrow \mathtt{DVD}^6 : \mathtt{t}'_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$. $\mathtt{DVD}^6 \rightarrow \mathtt{USER}^1 : \mathtt{s}'_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$. (12)

$\mathtt{recX}$.(

$\mathtt{USER}^1 \rightarrow \mathtt{DVD}^2 : \mathtt{s}_{\mathtt{DVD}}\langle\mathtt{play}\rangle$. (13)

$\mathtt{DVD}^2 \rightarrow \mathtt{TV}^3 : \mathtt{t}_{\mathtt{DVD}}\langle\mathtt{video\_on}\rangle$.

$\mathtt{TV}^3 \rightarrow \mathtt{LIGHT1}^4 : \mathtt{u}_{\mathtt{DVD}}\langle\mathtt{small}\rangle$.

$\mathtt{LIGHT1}^4 \rightarrow \mathtt{CURTAIN1}^5 : \mathtt{v}_{\mathtt{DVD}}\langle\mathtt{down}\rangle$.

$\mathtt{CURTAIN1}^5 \rightarrow \mathtt{LIGHT1}^4 : \mathtt{v}_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$. $\mathtt{LIGHT1}^4 \rightarrow \mathtt{TV}^3 : \mathtt{u}_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$.

$\mathtt{TV}^3 \rightarrow \mathtt{DVD}^2 : \mathtt{t}_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$. $\mathtt{DVD}^2 \rightarrow \mathtt{USER}^1\mathtt{w} : \mathtt{s}_{\mathtt{DVD}}\langle\mathtt{ack}\rangle.\mathtt{X}$ (14)

$\overset{\mathtt{p1}+\mathtt{p3}}{}$

$\mathtt{USER}^1 \rightarrow \mathtt{DVD}^6 : \mathtt{s}'_{\mathtt{DVD}}\langle\mathtt{stop}\rangle$. (15)

$\mathtt{DVD}^6 \rightarrow \mathtt{TV}^7 : \mathtt{t}'_{\mathtt{DVD}}\langle\mathtt{channel\_on}\rangle$.

$\mathtt{TV}^7 \rightarrow \mathtt{LIGHT1}^8 : \mathtt{u}'_{\mathtt{DVD}}\langle\mathtt{on}\rangle$.

$\mathtt{LIGHT1}^8 \rightarrow \mathtt{CURTAIN}^9 : \mathtt{v}'_{\mathtt{DVD}}\langle\mathtt{up}\rangle$.

$\mathtt{CURTAIN1}^9 \rightarrow \mathtt{LIGHT1}^8 : \mathtt{v}'_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$. $\mathtt{LIGHT1}^8 \rightarrow \mathtt{TV}^7 : \mathtt{u}'_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$.

$\mathtt{TV}^7 \rightarrow \mathtt{DVD}^6 : \mathtt{t}'_{\mathtt{DVD}}\langle\mathtt{ack}\rangle$. $\mathtt{DVD}^6 \rightarrow \mathtt{USER}^1 : \mathtt{s}'_{\mathtt{DVD}}\langle\mathtt{ack}\rangle.\mathtt{X}$ (16)

))

Figure 6. Global Description for DVD Service

5.2. Local Description with Priority

The syntax of the local calculus is given in Figure7. We borrow the idea of the priority guard from CPG by Phillips[2]. A priority guard is written as U:a, where U is a set of pairs of sessions and operators, Session names in U occurs free so that they can be bound by the session initiations.

The labelled transitions with the structural congruence are shown in Figure8. Here, we slightly extended the initialization communication so that the source appliance is immediately known. A prefix with a priority guard, $U : s \triangleright op(x).I$ becomes I after a communication in a session s with operation op , if the pair is not in U.

The reduction inside the appliance itself is expressed by $M \xrightarrow{\alpha}_U N$. U is removed when the communication takes place globally between the appliances. This shows that the priory is only considered within each appliance because we do not assume any global priory mechanism The communication between the appliances are expressed as $\xrightarrow{\alpha}$, without U.

$\mathtt{SYS} = \mathtt{C}_1 | \ldots | \mathtt{C}_n$ (17)

$\mathtt{C} ::= \mathtt{N}[\mathtt{A}]$ (18)

$\mathtt{A} ::= \mathtt{ch}(\mathtt{s}).\mathtt{A} \mid \overline{\mathtt{ch}}(\nu\, \mathtt{s}).\mathtt{A} \mid \mathtt{P}$ (19)

$\mathtt{P} ::= \mathtt{U} : \mathtt{s} \triangleright \mathtt{op}_i(\mathtt{x}_i).\mathtt{P}$ (20)

$\mid \mathtt{s} \triangleleft \mathtt{op}\langle\mathtt{j}\rangle.\mathtt{P}$ (21)

$\mid \mathtt{P} + \mathtt{Q}$ (22)

$\mid \mathtt{X}$ (23)

$\mid \mathtt{recX}.\mathtt{P}$ (24)

Figure 7. Syntax for the Local Description

Example

Local description for LIGHT1 is shown in Figure9. Adding the priority guard resolves the interference as shown in the local description. This description consists of two parts; (25) to (26) where the sessions are initiated in the respective channels which will be used by LIGHT1 during its interactions, and lines (27) to (30) show the actual communications of LIGHT1 The description (29) in Figure9 describes the behavior of LIGHT1 in Coming Home Service. We can observe a priority guard $(\mathtt{U}_{\mathtt{DVD}}, \mathtt{small})$ which indicates the session name and the operation name of the behaviour of LIGHT1 for DVD Play Service(I1), is written before the operation for Home Coming Service(I3). This shows that the *turn on* operation of I3(29) can not be able to operate if the *turn off* operation in I1 is still performing. This reflects the priority values where I1 has a value p1 and S3 has a value p2, and thus the interference can be resolved at the local behavior level.

$\mathtt{LIGHT1}[$

$\mathtt{ch}_{\mathtt{LIGHT1}}(\mathtt{u}_{\mathtt{DVD}}, \mathtt{TV}).\overline{\mathtt{ch}}_{\mathtt{CURTAIN1}}(\mathtt{v}_{\mathtt{DVD}}, \mathtt{LIGHT1}).\mathtt{v}_{\mathtt{DVD}} \triangleright \mathtt{ack}.\overline{\mathtt{u}}_{\mathtt{DVD}} \triangleleft \mathtt{ack}.$ (25)

$\mathtt{ch}_{\mathtt{LIGHT1}}(\mathtt{u}'_{\mathtt{DVD}}, \mathtt{TV}).\overline{\mathtt{ch}}_{\mathtt{CURTAIN1}}(\mathtt{v}'_{\mathtt{DVD}}, \mathtt{LIGHT1}).\mathtt{v}'_{\mathtt{DVD}} \triangleright \mathtt{ack}.\overline{\mathtt{u}'}_{\mathtt{DVD}} \triangleleft \mathtt{ack}.$

$\mathtt{ch}_{\mathtt{LIGHT1}}(\mathtt{u}_{\mathtt{IO}}, \mathtt{AIR\_CON}).\overline{\mathtt{ch}}_{\mathtt{PHONE}}(\mathtt{v}_{\mathtt{IO}}, \mathtt{LIGHT1}).\mathtt{v}_{\mathtt{IO}} \triangleright \mathtt{ack}.\overline{\mathtt{u}}_{\mathtt{IO}} \triangleleft \mathtt{ack}.$

$\mathtt{ch}_{\mathtt{LIGHT1}}(\mathtt{t}'_{\mathtt{DAY}}, \mathtt{LIGHTG}).\overline{\mathtt{ch}}_{\mathtt{LIGHT2}}(\mathtt{u}'_{\mathtt{DAY}}, \mathtt{LIGHT1}).\mathtt{u}'_{\mathtt{DAY}} \triangleright \mathtt{ack}.\overline{\mathtt{t}'}_{\mathtt{DAY}} \triangleleft \mathtt{ack}.$

$\mathtt{recX}$.((26)

$\mathtt{u}_{\mathtt{DVD}} \triangleright \mathtt{small}_{\mathtt{play}}.\overline{\mathtt{v}}_{\mathtt{DVD}} \triangleleft \mathtt{down}_{\mathtt{play}}.\mathtt{v}_{\mathtt{DVD}} \triangleright \mathtt{ack}.\overline{\mathtt{u}}_{\mathtt{DVD}} \triangleleft \mathtt{ack}.\mathtt{X}$ (27)

+

$\mathtt{u}'_{\mathtt{DVD}} \triangleright \mathtt{on}_{\mathtt{stop}}.\overline{\mathtt{v}'}_{\mathtt{DVD}} \triangleleft \mathtt{up}_{\mathtt{stop}}.\mathtt{v}'_{\mathtt{DVD}} \triangleright \mathtt{ack}.\overline{\mathtt{u}'}_{\mathtt{DVD}} \triangleleft \mathtt{ack}.\mathtt{X}$ (28)

+

$(\mathtt{u}_{\mathtt{DVD}}, \mathtt{small}_{\mathtt{play}}) : \mathtt{u}_{\mathtt{IO}} \triangleright \mathtt{on}_{\mathtt{in}}.\overline{\mathtt{v}}_{\mathtt{IO}} \triangleleft \mathtt{on}_{\mathtt{in}}.\mathtt{v}_{\mathtt{IO}} \triangleright \mathtt{ack}.\overline{\mathtt{u}}_{\mathtt{IO}} \triangleleft \mathtt{ack}.\mathtt{X}$ (29)

+

$(\mathtt{u}_{\mathtt{DVD}}, \mathtt{small}_{\mathtt{play}}) : \mathtt{t}'_{\mathtt{DAY}} \triangleright \mathtt{on}_{\mathtt{even}}.\overline{\mathtt{u}'}_{\mathtt{DAY}} \triangleleft \mathtt{on}_{\mathtt{even}}.$

$\mathtt{u}'_{\mathtt{DAY}} \triangleright \mathtt{ack}.\overline{\mathtt{t}'}_{\mathtt{DAY}} \triangleleft \mathtt{ack}.\mathtt{X}$)] (30)

Figure 9. Local Description for LIGHT1

6. Towards EPP with Priority

In the communication centered programming, EPP is the most important feature to develop reliable communicating

$$\mathtt{M}_1 + (\mathtt{M}_2 + \mathtt{M}_3) \equiv (\mathtt{M}_1 + \mathtt{M}_2) + \mathtt{M}_3 \quad (\mathtt{SUM}_{\mathtt{assoc}})$$
$$\mathtt{M}_1 + \mathtt{M}_2 \equiv \mathtt{M}_2 + \mathtt{M}_1 \quad (\mathtt{SUM}_{\mathtt{comm}})$$
$$\mathtt{M} + 0 \equiv \mathtt{M} \quad (\mathtt{SUM}_{\mathtt{zero}})$$
$$\mathtt{P}_1|(\mathtt{P}_2|\mathtt{P}_3) \equiv (\mathtt{P}_1|\mathtt{P}_2)|\mathtt{P}_3 \quad (\mathtt{PAR}_{\mathtt{assoc}})$$
$$\mathtt{P}_1|\mathtt{P}_2 \equiv \mathtt{P}_2|\mathtt{P}_1 \quad (\mathtt{PAR}_{\mathtt{comm}})$$
$$\mathtt{P}|0 \equiv \mathtt{P} \quad (\mathtt{PAR}_{\mathtt{zero}})$$
$$(\nu \mathtt{z})(\nu \mathtt{w})\mathtt{P} \equiv (\nu \mathtt{w})(\nu \mathtt{z})\mathtt{P} \quad (\mathtt{RES}_{\mathtt{comm}})$$
$$(\nu \mathtt{z})0 \equiv 0 \quad (\mathtt{RES}_{\mathtt{zero}})$$

$$(\mathtt{SUM}_{\mathtt{in}})\ \frac{}{\mathtt{M} + \mathtt{U} : \mathtt{s} \triangleright \mathtt{op}_i(\mathtt{x}_i).\mathtt{P} + \mathtt{N} \xrightarrow{\langle \mathtt{s}, \mathtt{op}_i, \mathtt{v} \rangle}_{\mathtt{U}} \mathtt{P}[\mathtt{v}/\mathtt{x}_i]}\ \text{if } \langle \mathtt{s}, \mathtt{op}_i \rangle \notin \mathtt{U}$$

$$(\mathtt{SUM}_{\mathtt{out}})\ \frac{}{\mathtt{M} + \mathtt{U} : \mathtt{s} \triangleleft \mathtt{op} < \mathtt{v} > .\mathtt{P} + \mathtt{N} \xrightarrow{\langle \overline{\mathtt{s}}, \mathtt{op}, \mathtt{v} \rangle}_{\mathtt{U}} \mathtt{P}}\ \text{if } \langle \overline{\mathtt{s}}, \mathtt{op}_i \rangle \notin \mathtt{U}$$

$$(\mathtt{RES})\ \frac{\mathtt{P} \xrightarrow{\alpha}_{\mathtt{U}} \mathtt{P}', \alpha \notin \{\langle \mathtt{s}, \mathtt{op}, \mathtt{v} \rangle, \langle \overline{\mathtt{s}}, \mathtt{op}, \mathtt{v} \rangle\}}{(\nu \mathtt{s})\mathtt{P} \xrightarrow{\alpha}_{\{\langle \mathtt{t}, \mathtt{op} \rangle, \langle \overline{\mathtt{t}}, \mathtt{op} \rangle \in \mathtt{U} | \mathtt{s} \neq \mathtt{t}\}} (\nu \mathtt{s})\mathtt{P}'} \qquad (\mathtt{REC})\ \frac{\mathtt{P}[\mathtt{recX}.\mathtt{P}/\mathtt{X}] \xrightarrow{\alpha}_{\mathtt{U}} \mathtt{P}'}{\mathtt{recX}.\mathtt{P} \xrightarrow{\alpha}_{\mathtt{U}} \mathtt{P}'} \qquad (\mathtt{COMP})\ \frac{\mathtt{P} \xrightarrow{\alpha}_{\mathtt{U}} \mathtt{P}'}{\mathtt{A}[\mathtt{P}] \xrightarrow{\alpha} \mathtt{A}[\mathtt{P}']}$$

$$(\mathtt{INIT})\quad \frac{}{\mathtt{ch}(\mathtt{s}, \mathtt{app}).\mathtt{P} \xrightarrow{\langle \mathtt{ch}, \mathtt{app}, \mathtt{t} \rangle} (\nu \mathtt{t})\mathtt{P}[\mathtt{t}/\mathtt{s}]} \qquad \frac{}{\overline{\mathtt{ch}}(\nu \mathtt{s}, \mathtt{app}).\mathtt{P} \xrightarrow{\langle \overline{\mathtt{ch}}, \mathtt{app}, \mathtt{t} \rangle} \mathtt{P}}\ \mathtt{t\ is\ a\ fresh\ name}$$

$$(\mathtt{COMM})\ \frac{\mathtt{A}_1[\mathtt{P}_1] \xrightarrow{\langle \mathtt{s}, \mathtt{op}, \mathtt{v} \rangle} \mathtt{A}_1[\mathtt{P}_1'], \quad \mathtt{A}_2[\mathtt{P}_2] \xrightarrow{\langle \overline{\mathtt{s}}, \mathtt{op}, \mathtt{v} \rangle} \mathtt{A}_2[\mathtt{P}_2']}{\mathtt{A}_1[\mathtt{P}_1] \mid \mathtt{A}_2[\mathtt{P}_2] \xrightarrow{\tau} \mathtt{A}_1[\mathtt{P}_1'] \mid \mathtt{A}_2[\mathtt{P}_2']} \qquad \frac{\mathtt{A}_1[\mathtt{P}_1] \xrightarrow{\langle \mathtt{ch}, \mathtt{app}, \mathtt{v} \rangle} \mathtt{A}_1[\mathtt{P}_1'], \quad \mathtt{A}_2[\mathtt{P}_2] \xrightarrow{\langle \overline{\mathtt{ch}}, \mathtt{app}, \mathtt{v} \rangle} \mathtt{A}_2[\mathtt{P}_2']}{\mathtt{A}_1[\mathtt{P}_1] \mid \mathtt{A}_2[\mathtt{P}_2] \xrightarrow{\tau} \mathtt{A}_1[\mathtt{P}_1'] \mid \mathtt{A}_2[\mathtt{P}_2']}$$

$$(\mathtt{L} - \mathtt{PAR})\ \frac{\mathtt{A}_1[\mathtt{P}_1] \xrightarrow{\alpha} \mathtt{A}_1[\mathtt{P}_1']}{\mathtt{A}_1[\mathtt{P}_1] \mid \mathtt{A}_2[\mathtt{P}_2] \xrightarrow{\alpha} \mathtt{A}_1[\mathtt{P}_1'] \mid \mathtt{A}_2[\mathtt{P}_2]} \qquad (\mathtt{CONG})\ \frac{\mathtt{M} \equiv \mathtt{M}', \mathtt{M}' \xrightarrow{\alpha}_{\mathtt{U}} \mathtt{N}', \mathtt{N}' \equiv \mathtt{N}}{\mathtt{M} \xrightarrow{\alpha}_{\mathtt{U}} \mathtt{N}} \qquad \frac{\mathtt{P} \equiv \mathtt{P}', \mathtt{P}' \xrightarrow{\alpha} \mathtt{Q}', \mathtt{Q}' \equiv \mathtt{Q}}{\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}}$$

Figure 8. Reduction Semantics for Local Description

programs. Although we have not fully developed the EPP with the priority, we examine the following two points. In [5], three basic principles for the global description have been identified, under which we can define a sound and complete EPP, *connectedness*, *well-threadedness*, and *coherence*. In the next section we will show whether the global description with priority satisfies three conditions. Further, we will show the equivalence of the global descriptions and local descriptions of the same service scenario.

6.1. Three principles for CDL

The global description for DVD Service is shown in Figure6. *Connectedness* dictates a local causality principle in interaction - if *A* initiates any action (say sending messages, assignment, ..) as a result of a previous event (e.g. reception of a message), then that preceding event should take place at *A*. A global description is *well-threaded* when it is connected and has a consistent annotation of threads. Finally, *coherence* is a consistency principle for the description of each participant in a global description that is connected and well-threaded. With the coherence property, nondeterministic choices over initial sessions do not introduce any nondeterminism. Just binding a initial session with a participant does not resolve the type of the session. It will be known by later communications. As it can be noticed, the participant are assigned with annotations of threads, which makes it obvious that the description satisfies the connectedness and the well-threadedness conditions. Moreover, observing the local descriptions of the appliances shows that the description satisfies the coherence principle.[2]

6.2. Correctness

In this section,we will explain the equivalence of the global description and the local description based on the examples. Figure10, Figure11 show the step-by-step computation of a same scenario; an example where a user at home is watching a DVD(I1) and another user comes home(I3). Figure10 shows the step-by-step computation of global descriptions of the scenario, and Figure11 shows the step-by-step computation of the local descriptions for the appliances, where both computations are based on the reduction semantics shown in Figure5, Figure8 respectively. Each

[2]Although we do not discuss the conditions for the remaining global descriptions and the local descriptions of the other appliances due to the limited space, the three conditions are already tested for those examples similarly.

step of the global description corresponds to that of the local description. According to the priority values we decided in section 4, the previous interaction(I1) has a higher priority that the later one(I3). Therefore, the light that was turned of by the DVD Play Scenario will not be turned on by the Home Coming Scenario, while the DVD is still playing. This can be observed from both descriptions.

7. Conclusion

In this paper, we presented the extended scheme for describing the interactions with priority in the home appliance network. In an SOA based network, where no centered single point of control exists, there is a need of a description for the distributed interactions where the flow can be observed globally. We found the communication centered programming framework fits well our purpose. In a small scale network like home appliance network, interference between interactions has to be resolved to keep the consistency of operations. To cope with the interference problem, we extended the global as well as the local descriptions by adding priority in our both descriptions. We confirmed the equivalence between two layered descriptions by hand. Moreover, we confirmed that the prioritized global description is well-structured. We presented an extension of descriptions in those two layers considering EPP with priority. In order to fully develop the precise EPP with priority, further investigation is required especially of typing the communications.

There are other researches that discuss the description for choreography. [12] discusses BPEL4Chor framework where participants' behavior as well as the participant declaration should be given by the user. However, in the case of autonomous-decentralized home appliance network, preparing the behavior for each appliance is not an easy task. Therefore, describing only a well-structured global view of the interactions and then deriving the local descriptions automatically is a user-friendly framework.

[11] discusses other solutions for resolving interference problem, such as prompting users when the interference in detected during runtime, assigning priority to the users or to the methods. As we take an approach for users to set up the scenario of the interaction in the system, assigning the priority to the interaction itself would be natural and a fine granularity for the interaction resolution in this framework.

We considered the three conditions, connectedness, well-threadedness, and coherence in [5] for the descriptions with priority as well. As a future research, we investigate the conditions of global descriptions in particular, with priority and the underlying theory of the local description with priority guard. To practically ease the programming integrated services, we further develop the support tools as done for WS-CDL[10]. At the moment, our focus is to enhance communication center programming by adding priority. This enables broader application of the approach. Having establishing the projection with priority, further formal development of actual technique to resolve interferences is a future work.

Acknowledgement

This research is partially supported by the Japanese Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C) 19500026, Scientific Research(B) 17300006.

References

[1] C.J. Fidge. A formal definition of priority in CSP. ACM Transactions on Programming Languages and Systems, 15(4):681-705, 1993.

[2] Iain Phillips. CCS with Priority Guards. In: Concur 2001, Lecture Notes in Computer Science 2154, pages 305-320, 2001. Springer-Verlag.

[3] J. Camilleri and G. Winskel. CCS with priority choice. Information and Computation, 116(1):26-37, 1995

[4] M. Kolberg, E.H. Magill, and M. Wilson. Compatibility Issues between Services supporting Networked Appliances, IEEE Communications Magazine, Vol. 41(11), 136-147, November 2003.

[5] Marco Carbone, Kohei Honda, Nobuko Yoshida Robin Milner, Gary Brown, Steve Ross-Talbot4 ” A Theoretical Basis of Communication-Centred Concurrent Programming. ” http://www.dcs.qmul.ac.uk/˜carbonem/cdlpaper/workingnote.pdf, 2006

[6] Marco Carbone, Kohei Honda, Nobuko Yoshida ” Structured Communication-Centred Programming for Web Services. ” http:// www. dcs. qmul. ac.uk/˜carbonem/cdlpaper/submission.pdf, ESOP 2007

[7] R.Cleaveland, G.Luttgen, and V.Natara jan. Priority in process algebra. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, Handbook of Process Algebra, chapter 12, pages 391-424. Elsevier, 2001.

[8] M.Nakamura, H.Igaki, H.Tamada and K.Matsumoto, ”Implementing integrated services of networked home appliances using service oriented architecture”, *Proc. of 2nd International Conference on Service Oriented Computing(ICSOC2004), pp269-278,Nov.2004.*

[9] Sakura Bhandari, Shoji Yuen, Kiyoshi Agusa, ”A Description for Integrated Operation of Network Appliances in Distributed Control Environment” (In Japanese) DSW 2007, pp35-44.

[10] Pi4 technology foundation: http://www.pi4tech.org/

[11] M. Nakamura, H. Igaki, K. Matsumoto, ”Feature Interactions in Integrated Services of Networked Home Appliances -An Object-Oriented Approach-,” Proc. of Int’l. Conf. on Feature Interactions in Telecommunication Networks and Distributed Systems (ICFI’05), pp.236-251, Jun. 2005.

[12] Decker, Gero Kopp, Oliver Leymann, Frank Weske, Mathias, ”BPEL4Chor: Extending BPEL for Modeling Choreographies”, Gero Decker et al, ICWS07.

$$\bullet \mathtt{IN-OUT_{SERVICE}} : \mathtt{USER} \rightarrow \mathtt{DOOR} : \mathtt{s_{IO}}\langle \mathtt{open}\rangle.\mathtt{I_1} \quad {}_{\mathtt{p2}}+_{\mathtt{p3}} \quad \mathtt{USER} \rightarrow \mathtt{DOOR} : \langle \mathtt{lock}\rangle.\mathtt{I_2},\ \sigma,\ \rho,\quad \mathtt{init} \tag{31}$$

$$\downarrow$$

$$\mathtt{IN-OUT_{SERVICE}} : \mathtt{USER} \rightarrow \mathtt{DOOR} : \mathtt{s_{IO}}\langle \mathtt{open}\rangle.\mathtt{I_1},\ \sigma,\ \rho,\quad \mathtt{p2}$$

$$\bullet \mathtt{IN-OUT_{SERVICE}} : \mathtt{USER} \rightarrow \mathtt{DOOR} : \mathtt{s_{IO}}\langle \mathtt{open}\rangle.\mathtt{I_1},\ \sigma,\ \rho,\ \mathtt{p2} \quad \textit{where} \quad \rho(\mathtt{USER}) = (\mathtt{p1}, \{\}), \rho(\mathtt{DOOR}) = (\mathtt{init}, \{\}) \tag{32}$$

$$\downarrow$$

$$\mathtt{IN-OUT_{SERVICE}} : (\mathtt{I_1},\ \sigma,\ \rho',\ \mathtt{p2}) \quad \textit{where} \quad \rho'(\mathtt{USER}) = (\mathtt{p1}, \mathtt{IN-OUT_{SERVICE}}), \rho'(\mathtt{DOOR}) = (\mathtt{p2}, \mathtt{IN-OUT_{SERVICE}})$$

$$\bullet \mathtt{IN-OUT_{SERVICE}} : \mathtt{DOOR} \rightarrow \mathtt{AIR\_CON} : \mathtt{t_{IO}}\langle \mathtt{on}\rangle.\mathtt{I_3},\ \sigma,\ \rho,\ \mathtt{p2}$$
$$\textit{where} \quad \rho(\mathtt{DOOR}) = (\mathtt{p2}, \mathtt{IN-OUT_{SERVICE}}), \rho(\mathtt{AIR\_CON}) = (\mathtt{init}, \{\}) \tag{33}$$

$$\downarrow$$

$$\mathtt{IN-OUT_{SERVICE}} : \mathtt{I_3},\ \sigma,\ \rho',\ \mathtt{p2} \quad \textit{where} \quad \rho'(\mathtt{DOOR}) = (\mathtt{p1}, \mathtt{IN-OUT_{SERVICE}}), \rho'(\mathtt{AIR\_CON}) = (\mathtt{p2}, \mathtt{IN-OUT_{SERVICE}})$$

$$\bullet \mathtt{IN-OUT_{SERVICE}} : \mathtt{AIR\_CON} \rightarrow \mathtt{LIGHT1} : \mathtt{u_{IO}}\langle \mathtt{on}\rangle.\mathtt{I_4},\ \sigma,\ \rho,\ \mathtt{p1}$$
$$\textit{where} \quad \rho(\mathtt{AIR\_CON}) = (\mathtt{p2}, \mathtt{IN-OUT_{SERVICE}}), \rho(\mathtt{LIGHT1}) = (\mathtt{p1}, \mathtt{DVD_{SERVICE}}) \tag{34}$$

$$\not\downarrow$$

Figure 10. Global flow of a scenario(A user comes home while an other user is watching the DVD)

$$\bullet \mathtt{USER}[\overline{\mathtt{s}}_{\mathtt{IO}} \lhd \mathtt{open}_{\mathtt{come}}.\mathtt{X} + \overline{\mathtt{t}}_{\mathtt{IO}} \lhd \mathtt{open}_{\mathtt{out}}.\mathtt{X} + ...\mathtt{X}] \mid \mathtt{DOOR}[\mathtt{s}_{\mathtt{IO}} \rhd \mathtt{open}_{\mathtt{come}}.\overline{\mathtt{t}}_{\mathtt{IO}} \lhd \mathtt{on}_{\mathtt{out}}.\mathtt{X} + ...\mathtt{X}] \mid \mathtt{AIR\_CON}[...] \mid \mathtt{LIGHT1}[...] \mid \mathtt{PHONE}[] \mid ....$$

$$\downarrow \tag{35}$$

$$\mathtt{USER}[\overline{\mathtt{s}}_{\mathtt{IO}} \lhd \mathtt{open}_{\mathtt{come}}.\mathtt{X} + \overline{\mathtt{t}}_{\mathtt{IO}} \lhd \mathtt{open}_{\mathtt{out}}.\mathtt{X} + ...\mathtt{X}] \mid \mathtt{DOOR}[\overline{\mathtt{t}}_{\mathtt{IO}} \lhd \mathtt{on}_{\mathtt{come}}.\mathtt{X}] \mid \mathtt{AIR\_CON}[...] \mid \mathtt{LIGHT1}[...] \mid \mathtt{PHONE}[] \mid ....$$

$$\bullet \mathtt{USER}[...] \mid \mathtt{DOOR}[\overline{\mathtt{t}}_{\mathtt{IO}} \lhd \mathtt{on}_{\mathtt{come}}.\mathtt{X}] \mid \mathtt{AIR\_CON}[\mathtt{t}_{\mathtt{IO}} \rhd \mathtt{on}_{\mathtt{come}}.\overline{\mathtt{u}}_{\mathtt{IO}} \lhd \mathtt{on}_{\mathtt{come}}.\mathtt{X} + ...\mathtt{X}] \mid \mathtt{LIGHT1}[...] \mid \mathtt{PHONE}[...] \mid ....$$

$$\downarrow \tag{36}$$

$$\mathtt{USER}[...] \mid \mathtt{DOOR}[\mathtt{t}_{\mathtt{IO}} \rhd \mathtt{open}_{\mathtt{come}}.\overline{\mathtt{u}}_{\mathtt{IO}} \lhd \mathtt{on}_{\mathtt{come}}.\mathtt{X} + ...\mathtt{X}] \mid \mathtt{AIR\_CON}[\overline{\mathtt{u}}_{\mathtt{IO}} \lhd \mathtt{on}_{\mathtt{come}}.\mathtt{X}] \mid \mathtt{LIGHT1}[...] \mid \mathtt{PHONE}[...] \mid ....$$

$$\bullet \mathtt{USER}[...] \mid \mathtt{DOOR}[...] \mid \mathtt{AIR\_CON}[\overline{\mathtt{u}} \rhd \mathtt{on}_{\mathtt{come}}.\mathtt{X}] \mid$$
$$\mathtt{LIGHT1}[\mathtt{u}_{\mathtt{DVD}} \rhd \mathtt{small}_{\mathtt{play}}...\mathtt{X} + \langle \mathtt{u}_{\mathtt{DVD}}, \mathtt{small}_{\mathtt{play}}\rangle : \mathtt{u}_{\mathtt{IO}} \lhd \mathtt{on}_{\mathtt{come}}.\overline{\mathtt{v}} \lhd \mathtt{play\_message}_{\mathtt{come}}.\mathtt{X} + ...\mathtt{X}] \mid \mathtt{PHONE}[...] \mid ...$$

$$\not\downarrow \tag{37}$$

Figure 11. Local flow of a scenario(A user comes home while another user is watching the DVD)

Performance Improvements for XML Security-Related Processing Using XML Parser Events

Takahide Nogayama[1], Toshiro Takase[1], Ken Ueno[1], Hyen-Vui Chung[2]

[1] *IBM Research, Tokyo Research Laboratory, 1623-14, Shimotsuruma, Yamato-shi, Kanagawa 242-8502, Japan*
{nogayama, E30809, kenueno}@jp.ibm.com
[2] *IBM Austin, 11400, Burnet Road, Austin, TX 78758-3415, United States of America*
hychung@us.ibm.com

Abstract

This paper describes performance improvements for a Web services engine. Tree-structured data models are used to represent XML data in many Web services engines. We can avoid traversing such data models and creating unneeded objects by doing XML related processing in the XML parser layer. We implemented this method on AXIOM. The AXIOM is a tree-structured data model used on Apache Axis2 which is currently the leading the next generation of Web services engine.

1. Introduction

Service Oriented Architecture (SOA) is one of the current new hot topics among IT professionals. However, SOA is not actually a new technology. SOA does not define a new technology, but rather is about integrating technical and business processes, as well as reuse and architecture. It is about how to design computer systems to be loosely coupled, with more agility to adopt new business requirements, and quicker to react to business requirement changes and integration. In theory, SOA does not require Web Services and CORBA (Common Object Request Broker Architecture). However, the trend of implementing SOA using Web Services is becoming prominent in the industry. According to one Gartner study [8], "Through 2008, SOA and Web services will be implemented together in more than 75 percent of new SOA or Web services projects (0.7 probability)." If this trend continues, Web Services will be an important technology in SOA.

Web Services have evolved over the year since SOAP 1.1 [1] was published in 2000. There are many specifications defining the Quality of Services (QoS) for Web Services. Security is one of the important QoS required to provide secure SOA-based applications. In 2002, IBM and Microsoft published a roadmap for Web services security. The specifications derived from the road map became an OASIS standard [11].

The Web services security standard (WS-Security) is currently inversion 1.1 [4]. It defines how to sign SOAP messages (based on the W3C XML digital signature recommendation [5]), how to encrypt SOAP messages content (based on the W3C XML encryption recommendation [6]), and how to propagate security tokens with the SOAP messages. This is an important QoS to provide secure and interoperable SOA environments.

Tree-based data models are widely used for representing XML data. Such models are used in many Web services engines to perform QoS. However creating tree-based data and processing the QoS with tree-based data has a high computational cost.

To solve the performance issues, Teraguchi et al. proposed the tree-based data cache [13]. It caches the subtrees of tree-based data and allows us to skip creating the subtrees that have already been created. Makino et al. proposed a template-based approach [14]. It extracts data without parsing the XML data and without creating tree-based data by matching templates. Template-based approach for SOAP was also proposed by Venkatesh et al. [15]. Makino et al. proposed a streaming WS-Security implementation [10]. It performs WS-Security processing in a SAX event stream without creating tree-based data.

Skipping creating a tree-based data is an efficient approach, but we have to create the tree-based data after all if other QoS components work with a tree-based data model. In this paper, we propose filtering with a streaming API. It performs the QoS processing on the streaming XML parser events. It is similar to Satoshi's implementation [10]. But our approach allows us to create tree-based data and support collaboration between the XML parser and a tree-based data builder.

We evaluated and compared the performance with our approach and with tree-based processing. AXIOM was used as the tree-based data model, since it is used in Axis2.

The rest of the paper is organized as follows: First, we introduce an overview of Web services engine and tree-based data models used in the Web services engine and point out the problems of the tree-based data models in Section 2. Then we propose an XML filtering for fast QoS processing in Section 3. Section 4 describes an application of XML filter for XML signature and WS-Security. Section 5 shows the experimental results with our method.

2. Details of Web Services Engine Process

This section describes how Web services engine processes and how its data model is constructed by an XML filter.

2.1. Main Components of Web Services and QoS-related Processing

A Web services engine is a software system providing Web services. Web services are a data transportation system over network protocol. In common cases, applications on the client side and server side communicate data as Extensible Markup Language (XML) messages using SOAP. The Web services engine combines the application and network components and supplies QoS for the Web services. Figure.1. illustrates components and data flow in Web services engine on the server side.

Web services use QoS for confidentiality, integrity, reliability, security, performance, and other reasons. In this paper, we focus on the canonicalization processing of XML signature and ID-related processing of WS-Security.

Handlers in Web services engine perform the QoS-related processing. A handler mechanism is defined in the JSR 109 specification [2]. The Object Model tree (OM tree) is used in the handler as XML data representation.

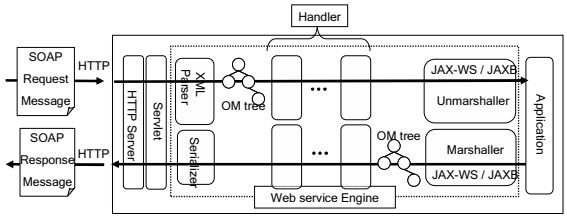


Figure 1. Main components of Web services on server side.

2.2. Tree-based Data Model

Tree-based data models are widely used for representing XML data. Such models are used in many Web services engines. DOM is the most popular tree-based data model. AXIOM is the tree-based data model which is used in Axis2.

Tree-based data models are good for representing XML data because XML uses tree structures and all data and text are stored in those structures. Such data models are easy for XML developers to use. In this paper, we call an instance of tree-based data in a runtime environment an OM tree. We call each node of an OM tree an OM node.

2.3. Streaming XML Parser and Builder

An XML parser is used for getting XML data from the text XML of an XML document. The DOM is using a tree-based API for accessing the XML data. We can access the XML data as if the XML data was stored in the tree. DOM is supported by Xerces, MSXML, and other parsers.

There are two types of event driven parser. We call them "stream-based XML parser." One is a "push" parser such as Simple API for XML (SAX) [9], and the other is a "pull" parser such as Streaming API for XML (StAX). We can also access XML data as an ordered sequence of events in document. An event represents an XML text node, an XML element node, an XML comment, and other XML features. An event is created when the parser encounters such XML features each time. Push parsers push the event into its application. The application has to catch the all of the events passed from the parser. On the other hand, for pull parsers, the event is pulled by applications. The applications can pull only required data. SAX is supported by Xerces, MSXML, and other parsers. MSXML also supports pull API. StAX is defined in JSR 173 [3] and supported by Woodstox.

If we want to use tree-based access with a streaming-based XML parser, it is necessary to create an OM tree from the sequence of events. For example,

in Axis2, the event processor (called “Builder”) is wrapped around the XML parser. If the Builder receives an event from the XML parser, the Builder creates an OM node, writes the event data into the OM node, and inserts the node into the OM tree (see Figure 2).

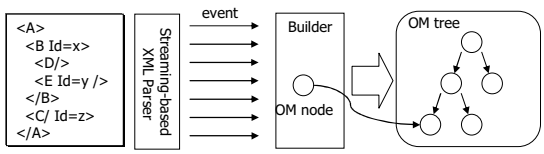


Figure 2. Streaming-based XML parser and builder

3. Design of XML Filter

This section describes issues of tree-based processing and design of our method XML filter.

3.1. Performance Issues of Tree-based Processing

XML-related processing with an OM tree becomes slow if it accesses the XML data repeatedly. For example, when we want to find an OM node that meets a certain condition, we must do breadth-first search or depth-first search in the OM tree. These kinds of searches from the root node of the tree are done for each request to find an OM node. Thus we have to search the same paths and examine the same OM nodes several times.

Building an OM tree generally has a high computational cost because of the amount of memory that must be allocated for each OM node.

3.2. XML Filter

To solve the performance issues, we propose filtering for a streaming API. We call this XML filtering. The concepts of filters have been used for many years in the fields of image processing, signal processing, and other areas. An XML filter has many of the same features as these filters.

An XML filter can be implemented in a streaming XML API such as SAX or StAX and so on. Normally the Builder links to a streaming XML parser. In this approach, one or more XML filters can be inserted between the Builder and the streaming-based XML parser forming a chain (see Figure 3). By definition a parent XML filter is upstream of a child XML filter. If a parent XML filter or builder invokes the streaming XML API for an XML filter, then that XML filter invokes the child XML filter.

An XML filter can monitor or change the XML data by intercepting the streaming XML parser events. If the XML filter is a data conversion filter, the XML filter changes the returned value from the child XML filter, and returns the values to the parent XML filters or builder.

An XML filter can also works directly with the Builder. The Builder has a reference to the OM node that is being created for the current event. If some types of OM nodes are frequently accessed, then we should use an OM node list because traversing the OM tree has a high computational cost. We can use an XML filter to create the OM node list in the building phase. Such an XML filter checks whether the current event satisfies the condition or not, and if so, the XML filter gets the current OM node from the Builder and adds it to the OM node list. After building, we can access the OM node without tree traversal by using the list.

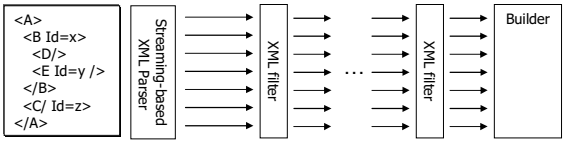


Figure 3. XML filter chain

4. XML Filter for XML Signature and ID Reference

This section describes an application of XML filter for XML signature and ID reference of WS-Security. XML filters are implemented for StAX.

4.1. Example of Signed SOAP Message

An example of a signed SOAP message is shown in Figure 4. In this example, the SOAP body, from <soapenv:Body> to </soapenv:Body>, is signed.

The signature information as processed by sender is stored in <ds:Signature>. The receiver verifies the signature by using the information in this element. The value of the attribute "URI" in <ds:Reference> indicates which element is signed. The value of the attribute "Algorithm" in <ds:Transform> describes the canonicalization.

The application data is stored in <soapenv:Body>. The value of the attribute "wsu:Id" in <soapenv:Body> is used to access this element by using the ID reference defined in WS-Security.

```
<soapenv:Envelope
   xmlns:soapenv="..." xmlns:wsu="...">
 <soapenv:Header>
  <wsse:Security xmlns:wsse="...">
   <wsse:BinarySecurityToken
    EncodingType="..." ValueType="..." wsu:Id="id_2" >
     ...
   </wsse:BinarySecurityToken>
   <ds:Signature xmlns:ds="...">
    <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="..." />
     <ds:SignatureMethod Algorithm="..." />
     <ds:Reference URI="#id_1">
      <ds:Transforms>
       <ds:Transform Algorithm="..." />
      </ds:Transforms>
      <ds:DigestMethod Algorithm="..." />
      <ds:DigestValue>...</ds:DigestValue>
     </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>...</ds:SignatureValue>
    <ds:KeyInfo>
     <wsse:SecurityTokenReference>
      <wsse:Reference  URI="#id_2"  ValueType="..."
      />
     </wsse:SecurityTokenReference>
    </ds:KeyInfo>
   </ds:Signature>
  </wsse:Security>
 </soapenv:Header>
 <soapenv:Body wsu:Id="id_1" xmlns:wsu="...">
  <TradeRequest>
   <Price>1000</Price>
  </TradeRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Figure 4. A signed SOAP message

4.2. XML Signature, ID Reference and Tree-based Processing

Many web services engine perform XML signature and ID-related processing with tree-based data model. We show how it is performed here.

4.2.1. ID Reference and ID Uniqueness of WS-Security

An ID reference is used when there is need to refer to an element such as for signature references. We can identify an element by using the value of the attribute "wsu:Id." The specification also states that an XML document must not have the same value of "wsu:Id." Checking that all of ID values differ in the document is called "an ID uniqueness check".

To get an element by using an ID reference and to check the ID uniqueness, when using tree-based processing, we have to traverse all of the OM nodes in the tree data (see Figure 5).

4.2.2. Canonicalization of XML Signature

Signature verification is done by comparing the digest value of the signed element to the digest value attached by sender. Before calculating the digest value, the signed element is transformed to a byte array. Exclusive canonicalization (C14N) [7] is generally used for this transformation. Exclusive canonicalization is one of the XML conversion algorithms. If the canonical form of two XML documents are equivalent, then both are logically the same.

To get the canonical form from an OM tree, we traverse the signed element in depth-first order and write the canonical form of the OM nodes in the subtrees to the byte array (see Figure 6).

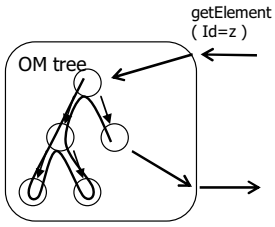


Figure 5. An ID reference with tree-based processing

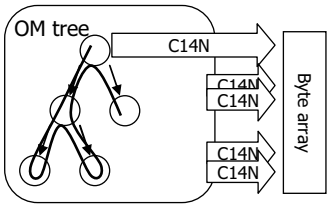


Figure 6. Canonicalization with tree-based processing

4.3. XML filter-based Canonicalization of XML Signature and ID Reference

We show XML filters for XML signature and ID reference here. These XML filters are implemented for StAX and AXIOM.

4.3.1. XML Filter for StAX

In this paper we implemented XML filters for StAX. We used StAXBuilder from AXIOM.

In a normal case, StAXBuilder invokes next() from the StAX parser. StAX parser returns an event type. StAXBuilder creates an appropriate OM node according to the event type, gets the information for the event from the StAX parser by using one of the getXXX() methods (e.g. getLocalName() and getNamespaceURI() and so on), writes the information into the OM node, and attaches the OM node to the OM tree.

When an XML filter is used, one or more XML filters are inserted between StAXBuilder and the StAX parser. When StAXBuilder invokes next() with an XML filter, that XML filter invokes next() from the StAX parser and gets the event type of the current event. The XML filter does its own processing (such as ID-related processing or canonicalization processing as described in next subsection) according to the event type. Then the XML filter returns the event type to StAXBuilder. StAXBuilder gets information about the event to build the OM node by using getXXX() methods. If the purpose of the XML filter is read only such as for logging, then the XML filter performs no visible actions. If the XML filter needs to change the returned values, then the XML filter changes the values and returns the changed values to StAXBuilder.

4.3.2. ID Filter and ID-Node Map

The purpose of an ID filter is creating an ID-Node map. The ID-Node map is a hash map. The ID-Node map has all of the pairs of Id value and reference for each OM node in the document. When we want to get a reference of an OM node which has a certain Id value, we can get the reference by passing the Id value to the ID-Node map.

An ID filter is inserted between StAXBuilder and the StAX parser. The ID filter monitors all StAX events. If the event type is a start element, the ID filter searches for an Id attribute. If the Id attribute exists, the ID filter takes the Id value and gets a reference to the OM node which is being created for the current event by StAXBuilder. Then the ID filter records the pair of the Id value and the reference into the ID-Node map.

The ID uniqueness check is also performed in this building phase. When the ID filter records the pair, it also checks to see if that Id value exits in ID-Node map, and if so, the ID filter knows that (at least) two Id attributes have the same Id values. If the tree building is completed with no duplications, then the ID filter knows that all of the Id attributes have unique values.

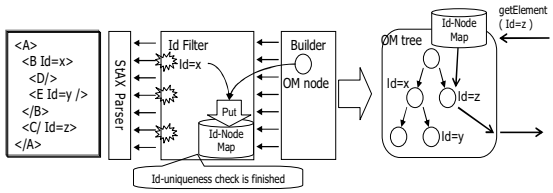


Figure 7. An ID reference processing with an ID filter

4.3.3. Canonicalization Filter

The purpose of a canonicalization Filter (C14N filter) is to create the canonical form of any element.

A C14N filter is inserted between StAXBuilder and the StAX parser by a security header filter described in next subsection. The security header filter gives the Id value of the signed element to the C14N filter. First, C14N filter searches a start element of the signed element. C14N filter monitors the event types of the StAX events. If the event type is a type of start element, the C14N filter searches for an Id attribute. If an Id attribute exists in the event and the Id value is same as the specified Id value from the security header, C14N filter switches its mode from searching to doing canonicalization. From the start element to the end element of the signed element, the C14N filter writes the canonical form for the each StAX event to a byte array. At the end of the signed element, the C14N filter writes the completed canonical form, calculates the digest value of the byte array.

The handler verifies the signature by using the digest value.

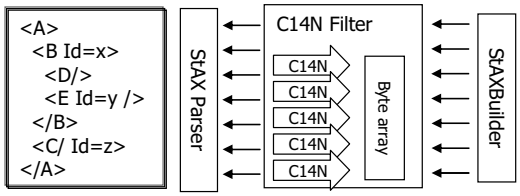


Figure 8. Canonicalization with a C14N filter

4.3.4. Security Header Filter

The purpose of a security header filter is to insert C14N filter into an XML filter chain. The security header filter watches for the StAX event of s security header. If the security header filter finds a <ds:Signature> tag, it inserts a C14N filter, stores the Id value of the signed element into the C14N filter and specifies the digest algorithm for the C14N filter.

A security header filter is also inserted by the ID filter if the ID filter finds a <wsse:Security> tag.

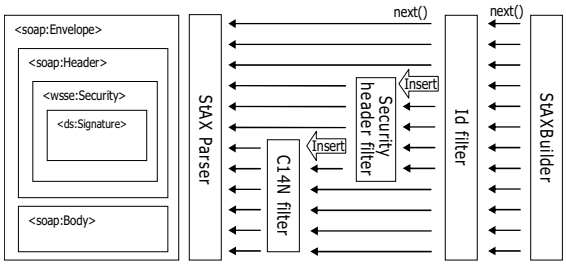


Figure 9. Overview of XML filter chain for SOAP body signed message

4.3.5. Limitations of XML Filter

XML Filter can not process previous events. If a security header filter is going to insert a C14N filter, and if the target signed element is an ancestor element, then the events for the ancestor element have already been consumed. We call this a backward ID reference.

Since order of children of security header, <wsse:Security>, is designed to process in document order, most of ID references for WS-Security are forward ID reference.

However, we can detect backward ID reference. If a target Id is already in ID-Node map, then the target signed element must be an ancestor element. If a security header filter detects a backward ID reference, it should not insert a C14N filter for that target. In this case, we have to use tree-based C14N processor instead of C14N filter.

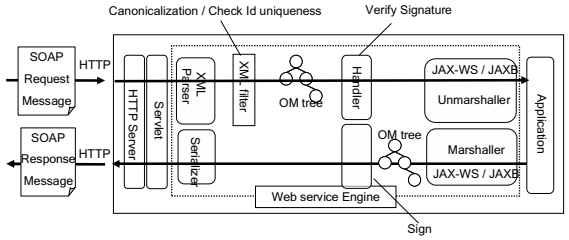


Figure 10. Delegation from handlers to XML filters

5. Performance Evaluation

We validated the benefit of our approaches by measuring and comparing the performance with the C14N library in IBM XML Security Suite for Java (XSS4J) and the tree-based ID processing. (Note that XSS4J is distributed on alphaWorks [12].) Since the XSS4J works with DOMs, we extended it to work with AXIOM.

We measured the performance for building complete OM trees, canonicalizing SOAP bodies, ID processing (referring for each request elements in the SOAP body and to check the ID uniqueness) for the example in Figure 4.

We varied the number of request elements in the SOAP body from 1 to 16. One request element was about 3,000 ASCII characters long. Each request element had a unique Id attribute, and included 73 start elements, 73 end elements, 50 text elements and 5 namespace URI declarations. The average prefix length was close to 3 characters. The average local name length was approximately 11 characters. The average text length of the text in a text element was approximately 9 characters.

The test code accepts as its input a Java InputStream object that represents the SOAP message. The input is parsed by the StAX parser and the OM tree is created by the AXIOM StAXBuilder of AXIOM. Then we canonicalized by using XSS4J and our method. We retrieved the OM node for all of the requests in the SOAP body by using the ID reference and checked the ID uniqueness using tree-based processing and our method.

We measured the elapsed time for the test code over 100,000 iterations. Before the measurement, we did 100,000 warm-up iterations. The results of the measurements are shown in Figure 11.

The performance measurements for this research were conducted using IBM xSeries 460 (64bit Intel Xeon Processor MP, 3.33GHz x 4 CPUs, 1MB L2 Cache, 8MB L3 Cache, 16GB RAM). The operating system on the experimental computer was Microsoft Windows Server 2003 Enterprise x64 Edition SP1. The performance measurement test code was run in the IBM J9 Java 2 Runtime Environment Standard Edition (IBM J9 JDK 5.0 SR4). During performance test runs, we pinned JVM process to one of four CPUs. We used AXIOM 1.2.4 and Woodstox 3.1.1 as the StAX XML parser.

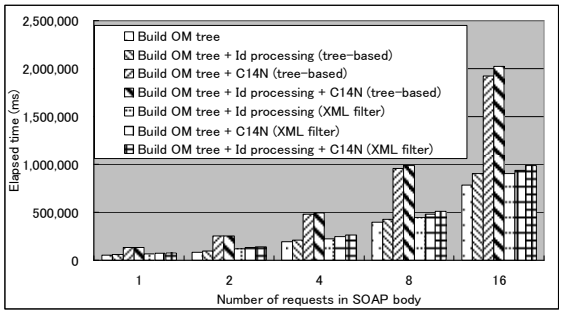


Figure 11. Performance comparisons between XML-filter-based processing and tree-based processing (lower is better)

For C14N, an XML filter was faster than tree-based processing for all message sizes. As regards ID processing, an XML filter was faster than tree-based processing for the large messages. Tree-based ID processing was faster than the XML filter for the small messages. If we did both C14N and ID processing, XML filter was faster than the tree-based processing for all message sizes.

C14N filter was significantly faster than tree-based processing. One of the reasons is that the C14N filter does not have to prepare information of the scope of the namespaces. When the C14N filter canonicalizes an event, An XML filter has the information for the event.

The C14N filter just asks it to the XML parser at that event. On the other hand, tree-based processing needs to prepare information of the scope of the namespaces because we can not ask it to the XML parser. To prepare the information of the scope of the namespaces from OM tree, we have to gather all namespace declarations of the signed element and all of its descendants.

We found that introducing an XML filter imposes a certain performance cost. However, the cost was almost the same if we introduced two or more XML filters because the elapsed time increases for both C14N and ID processing are smaller than the sum of the elapsed time increases for applying them separately. For both C14N and ID processing, our method is 40%-51% faster than the tree-based approach.

6. Conclusion

In this paper, we proposed a framework for XML filters to perform XML-related processing in a streaming XML parser layer. We showed that this framework can attain good performance compared to processing on tree-based data. Since introducing the XML filter has a certain performance cost, the higher benefits are when two or more XML filters are needed.

We focused on C14N in this paper because C14N is one of big performance overhead in WS-Security processing. But reducing only C14N cost is not enough to make whole WS-Security processing fast.

For the future work, we want to explore XML decryption, as specified in the XML Encryption Syntax and Processing [6]. We anticipate that using an XML filter will result in good performance with a reduced memory allocation. If a decryption filter receives encrypted events, it can decrypt and discard the encrypted events, and then, emit the decrypted events. Thus we should be able to produce a decrypted OM tree without ever building an encrypted OM tree.

References

[1] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. "Simple Object Access Protocol (SOAP) 1.1," W3C Note, May 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[2] Jim Knutson, and Heather Kreger. JSR 109: Web Services for J2EE 1.0 Final Release, Sep. 21, 2002. http://jcp.org/en/jsr/detail?id=109

[3] Christopher Fry. JSR 173: Streaming API for XML. Java Community Process Specification Final Release, Mar. 25, 2004. http://jcp.org/en/jsr/detail?id=173

[4] Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, and Ronald Monzillo. Web Services Security: SOAP Message Security 1.0 (WS-Security 2004), OASIS Standard 200401, Mar. 2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[5] Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, Ed Simon. "XML-Signature Syntax and Processing," W3C Recommendation, Feb. 2002. http://www.w3.org/TR/xmldsig-core/

[6] Takeshi Imamura, Blair Dillaway, Ed Simon. "XML Encryption Syntax and Processing," W3C Recommendation Dec. 2002. http://www.w3.org/TR/xmlenc-core/

[7] John Boyer, Donald E. Eastlake 3rd, Joseph Reagle, "Exclusive XML Canonicalization Version 1.0," W3C Recommendation, Jul. 2002. http://www.w3.org/TR/xml-exc-c14n/

[8] Gartner Research: Service-Oriented Architecture: Mainstream Straight Ahead by David W. McCoy and Yefim V. Natis. Publication date: 16 Apr. 2003 (LE-19-7652).

[9] David Brownell: SAX2, O'Reilly, ISBN 0-596-00237-8 W. Scott Means, Michael A. Bodie: The Book of SAX, No Starch Press, ISBN 1-886411-77-8.

[10] Satoshi Makino, Kento Tamura, Takeshi Imamura and Yuichi Nakamura, “Implementation and Performance of WS-Security”, International Journal of Web Services Research. Vol. 1, No.1, Mar. 2004.

[11] Organization for the Advancement of Structured Information Standards (OASIS). http://www.oasis-open.org/home/index.php

[12] alphaWorks. http://www.alphaworks.ibm.com/

[13] Masayoshi Teraguchi, Satoshi Makino, Ken Ueno, Hyen-Vui Chung, “Optimized Web Services Security Performance with Differential Parsing”, In Proceedings of the 2006 International Conference on Service Oriented Computing (ICSOC 2006), pages 277-288, Dec. 2006.

[14] Satoshi Makino, Michiaki Tatsubori, Kent Tamura, Yuichi Nakamura: “Improving WS-Security Performance with a Template-Based Approach”, In Proceedings of the 2005 IEEE International Conference on Web Services (ICWS 2005), Industrial Session, Orland, Florida, USA, July 11-15, 2005, pp.581-588, July. 2005.

[15] D. Andresen, D. Sexton, K. Devaram, and V. P. Ranganath. LYE: A High Performance Caching SOAP Implementation. In Proceedings of the 2004 International Conference on Parallel Processing (ICPP 2004), pages 143-150, Aug. 2004.

An Efficient DOM Implementation based on Literal XML Representation for SOAP Intermediary

Toshiro Takase[1, 2], Keishi Tajima[1]
[1]*Department of Social Informatics, Kyoto University,*
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
[2]*IBM Research, Tokyo Research Laboratory,*
1623-14, Shimotsurima, Yamato-shi, Kanagawa-ken 242-8502, Japan
E30809@jp.ibm.com, tajima@i.kyoto-u.ac.jp

Abstract

In practical Web services deployment, there are some SOAP intermediaries in the network that mediate the SOAP messages. The intermediaries add some support services to the SOAP message exchange, such as routing, logging, and security. The typical processing by a SOAP intermediary is parsing the SOAP messages, checking the data in each message, and serializing the messages to back it into the network. DOM is a very popular interface to navigate an XML message. Existing DOM implementations are not efficient for processing in a SOAP intermediary. Existing DOM implementations parse to tree data models and traverse the tree models for serialization. Typically, a SOAP intermediary does not modify the tree data model itself. In this situation, the creation of the tree data model and the traversal of the tree data is verbose. We propose a DOM implementation based on a literal XML representation. In our implementation, a SOAP intermediary does not have to traverse all of the tree data during serialization. We prototyped the DOM implementation and evaluated its performance.

1. Introduction

The service-oriented architecture (SOA) is being widely adopted in recent years. SOA is a paradigm for organizing and utilizing distributed capabilities that may be controlled and owned by different domains.

In practice, Web services technologies such as XML [2] and SOAP [3] are used to build distributed SOA system. In a real network, there are many proxies to provide various functions such as routing, logging, or security. Proxies in the SOAP layer are called SOAP intermediaries. The performance of a SOAP intermediary can be a problem because the processing on the SOAP intermediary may include heavy XML processing. In this paper, we propose a method for efficient XML processing on SOAP intermediaries.

The rest of this paper is structured as follows: First, we describe the XML processing at a SOAP intermediary in Section 2. Section 3 discusses design considerations for our efficient DOM implementation. Our implementation's details are presented in Section 4. Section 5 shows the results of our experiments. We discuss future work in Section 6. Finally, in Section 7 we conclude the paper.

2. XML processing at SOAP intermediaries

The term "SOAP intermediary" is defined in the SOAP 1.2 specification [3]. A SOAP intermediary is both a SOAP receiver and a SOAP sender and is targetable from within a SOAP message. It processes the SOAP Header blocks targeted at it and acts to forward a SOAP message towards an ultimate SOAP receiver.

The intermediary provides routing, logging, security-related processing, or other services. For the processing, the intermediary usually checks the SOAP messages but does not change the messages. Also, the intermediary often reads only the SOAP Header blocks, and it does not access the SOAP Body part.

Figure 1 represents a data transition on a SOAP intermediary. In the processing on the SOAP intermediary, an application on the intermediary has to do XML parsing of the incoming SOAP messages to read them. Then the application does some processing. Finally, the intermediary should serialize the SOAP message to send the message to the next destination. The XML parsing and the XML serialization are

reverse data conversion. In many cases, this reverse conversion on the SOAP intermediary is verbose.

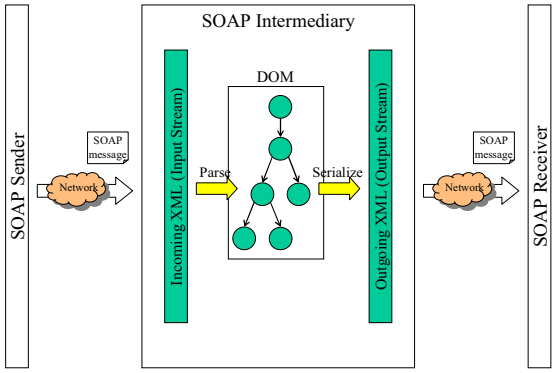


Figure 1. Data transitions on a SOAP intermediary

There are several types of XML parsing. The two most popular APIs are for DOM and SAX. The DOM [4] parsing creates a tree-based data model from an XML document. The tree data model, a DOM tree, is easy for developers to handle. The DOM tree, however, consumes lots of memory because all of an incoming XML document is converted into a DOM tree. In contrast, SAX [5] is an event driven parser API. The SAX parser sequentially notifies applications about the XML events. A SAX parser does not use much memory, but it is more difficult to use.

XSLT [8] processing may be used on a SOAP intermediary because XSLT can transform XML messages. On many SOAP intermediaries, however, incoming XML messages are not changed, but just checked for purpose such as routing or logging. For these purposes, XSLT may not be required.

Some XML appliances do XSLT transformations and some routine functions very quickly. This is very useful, but developers may want to build their own original logic on their SOAP intermediaries.

Overall, DOM is easy for developers to work with. Therefore, in this paper, our target is a DOM implementation on a SOAP intermediary. We propose a new efficient DOM implementation based on literal XML representation, thus reducing the redundancy of the paired of parsing and serialization on the SOAP intermediary.

3. Design consideration

In this section, we discuss design considerations with respect to full DOM versus partial DOM parsing, and read-only vs. read-write access.

3.1. Full DOM vs. Partial DOM parsing

In typical processing on SOAP intermediaries, only small parts of the messages are accessed. This means the entire XML document does not need to be parsed. In particular, in SOAP intermediaries usually only the SOAP Header is accessed. A SOAP message consists of one SOAP Envelope, which has one or zero SOAP Headers followed by one SOAP Body.

The StAX parser [6] uses another event-driven parser API. SAX is called a “push” parser, while StAX is “pull” parser. An application pulls each event from the StAX parser. The application also can stop the parse in the middle. By using a StAX parser, we could read just the SOAP Header parts.

Lazy DOM [1], and the defer-node-expansion option in Apache Xerces [7] are for lazy DOM parsing. In parsing, they create their own internal representation, and a partial DOM tree is created for just the accessed parts. In serialization, however, all of the XML is accessed for the traversal. Therefore, this approach is useless for SOAP intermediaries.

In this paper, we use normal full DOM parsing for simplicity. We discuss partial DOM parsing for SOAP intermediaries in the future work section.

3.2. Read-only vs. Read-Write access

In typical processing on SOAP intermediaries, incoming XML messages are rarely modified. If the incoming XML message is not changed in SOAP intermediary, the incoming literal XML representation is the same as the serialized form for the outgoing message. If the incoming XML message is changed, then the incoming literal representation cannot be used as it is for the serialized form.

In this work, our focus is cases where the incoming XML messages are not changed. Our DOM implementation preserves the incoming literal XML representation, and uses it as the serialized form. If the incoming XML message is modified, then the literal XML representation is discarded. By using a dirty flag, we can know if an XML message has been changed.

We may be able to reflect changes to a DOM into the literal XML representation. In the future work section, we discuss this possibility.

4. Implementation

In this section, we present the details of our DOM implementation. Here, we call our DOM the “Literal

DOM." Figure 2 shows the data structure of the Literal DOM. We prototyped our Literal DOM in Java.

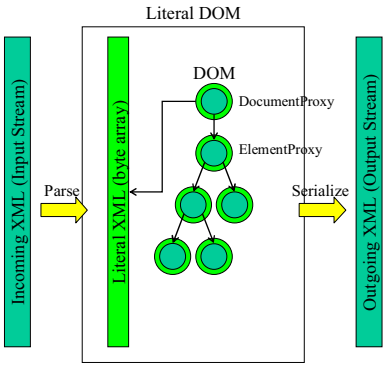


Figure 2. Data structure of the Literal DOM

An existing DOM implementation was used as the base of our Literal DOM implementation. The Literal DOM wraps the original DOM tree. For example, the `ElementProxy` object of the Literal DOM wraps the original `Element` object. The `ElementProxy` class implements all methods of the `Element` interface. Each read method just delegates an original method. A write method, such as `Element.appendChild()`, delegates and sets the dirty flag to true.

A `DocumentProxy` object (wrapping an original `Document` object) internally holds the incoming Literal XML as a byte array. In serialization, if the dirty flag is false, then the Literal XML is used as it is for the serialized form, and otherwise the DOM tree is traversed to serialize the tree.

The incoming XML may use various character encodings. Because the byte array of the Literal XML in our DOM is the same as the raw incoming XML, the character encoding is also same as the incoming XML. If we would like to serialize in some other character encoding, then we have to serialize by traversing the DOM tree. However this is a rare case.

Our Literal DOM implementation is compliant with the DOM specification. Therefore, existing applications on SOAP intermediaries do not need any modifications.

5. Performance evaluation

In this section, we evaluate the performance gain from our Literal DOM implementation. In this experiment, the Apache Xerces [7] XML parser was used as the base DOM implementation. Xerces is one of the most popular open source XML parsers in Java.

Figure 3 shows the parsing and serializing times of a SOAP message. The values in the graph are elapsed times for 100,000 iterations. The taller bars are slower. The experimental environment is Apache Xerces 2.9.0, the Sun Hotspot Server VM 1.5.0, Windows XP Professional SP2, and a ThinkPad T60 (Intel Core 2 Duo T7200 2.0 GHz) with 2.0 GB memory.

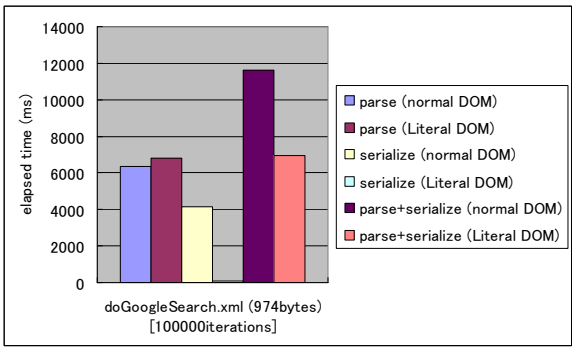


Figure 3. Performance of parsing and serialization for normal DOM and Literal DOM

The graph shows the performance of parsing, serialization, and combined parsing and serialization for the normal DOM (Xerces) and our Literal DOM. The parsing time of the Literal DOM is a little slower than the parsing time of the normal DOM. This is due to the overhead of wrapping the original DOM tree objects and preserving the literal XML as a byte array. The serialization time of the Literal DOM is extremely small. Actually, the serialization of the Literal DOM is just copying the literal XML byte array. Finally, with respect to the combined parsing and serialization, our Literal DOM is approximately 1.7 times faster than the normal DOM.

6. Future work

In this section, we discuss some future work. Most importantly, and as we mentioned in Section 3.1, we could implement partial DOM parsing.

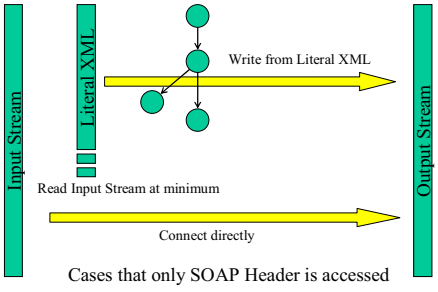


Figure 4. Hybrid between stream and object model

Figure 4 shows a possible design for partial DOM parsing. The DOM tree would be created only for the accessed part. This might be implemented by using the StAX parser, or we might implement it by ourselves. Since the StAX parser can be stopped, only the SOAP Header parts could be read and a DOM tree for the SOAP Header parts would be created. Alternately, we might be able to implement using Lazy DOM with Literal XML. In this case, the Literal XML representation should be referred to as internal data.

In addition, it might be possible to avoid reading the input stream of incoming XML messages. Only the accessed parts of the input stream actually need to be read, and then the rest of the input stream could be connected directly to the output stream for the outgoing XML.

In this paper, we assume that the incoming XML messages are rarely modified in the SOAP intermediary. However, even if the incoming messages are slightly changed, then the changes could be reflected efficiently.

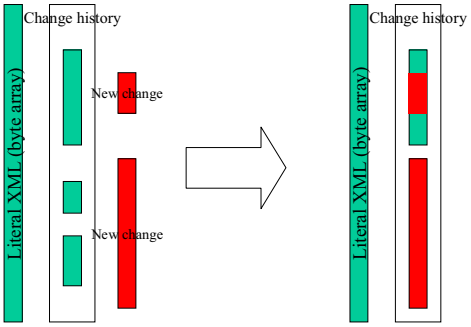


Figure 5. Change history for Literal XML

It may be inefficient to immediately reflect the changes to the DOM into the Literal XML. By using a change history, the actual change to the Literal XML could be deferred until the XML is serialized. Figure 5 shows a possible design with a change history for the Literal XML. Because XML is a tree structure, a new change either covers old changes or a new change is included in an old changed region. Due to this characteristic of XML, each change in the change history can be managed as one sequence in the document order. In serializing the Literal XML with some changes, the sequence of the document order could efficiently reflect these changes.

7. Concluding remarks

In this paper, we proposed a new DOM implementation to optimize serialization. The implementation is based on a literal XML representation. In typical processing on a SOAP intermediary, only the SOAP Header parts are accessed, and the incoming SOAP messages are not modified. In such cases, the pair of XML parsing and XML serialization is redundant. We implemented a prototype of our Literal DOM. We experimented with our prototype, showing the performance was faster than an existing DOM implementation.

References

[1] M. L. Noga, S. Schott, and W. Lowe. “Lazy XML Processing.” ACM Press, Proc. of the ACM Symposium on Document Engineering (DocEng) 2002.

[2] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. “Extensible Markup Language (XML) 1.0 (Fourth Edition),” W3C Recommendation 16 August 2006, http://www.w3.org/TR/xml/

[3] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon. “SOAP Version 1.2 Part 1: Messaging Framework (Second Edition),” W3C Recommendation 27 April 2007, http://www.w3.org/TR/soap12-part1/

[4] A. Hors, P. Hegaret, L. Wood, G. Nicol, J. Robie, M. Champion, and S. Byrne. “Document Object Model (DOM) Level 3 Core Specification,” Version 1.0, W3C Recommendation 07 April 2004, http://www.w3.org/TR/DOM-Level-3-Core/

[5] SAX, Simple API for XML. http://www.saxproject.org/

[6] JSR 173: StAX: Streaming API for XML. http://jcp.org/en/jsr/detail?id=173

[7] The Apache Software Foundation. Apache Xerces. http://xerces.apache.org/

[8] Michael Kay, “XSL Transformations (XSLT) Version 2.0,” W3C Recommendation 23 January 2007, http://www.w3.org/TR/xslt20/

ISBN 978-4-915256-70-7 C3040

Publisher :
Information Processing Society of Japan
Kagaku-kaikan (Chemistry Hall) 4F
1-5 Kanda-Surugadai, Chiyoda-ku, Tokyo 101-0062 JAPAN
Tel:+81-3-3518-8374 Fax:+81-3-3518-8375