A UML Model-Driven Business Process Development Methodology for a Virtual Enterprise using SOA & ESB

Sam Chung[1], Sergio Davalos[2], Craig Niiyama[1]
[1]Institute of Technology/[2]Milgard School of Business
University of Washington, Tacoma
Tacoma, Washington, USA
{chungsa, sergiod, cniiyama}@u.washington.edu

Daehee Won, Seung-Ho Baeg, Sangdeok Park
Division of Applied Robot Technology
Korea Institute of Industrial Technology
Ansan, Gyeonggi-Do, Korea
{daehee, shbaeg, sdpark}@kitech.re.kr

Abstract—**The purpose of this paper is to demonstrate how a Virtual Enterprise Integration (VEI) project using Service-Oriented Architecture (SOA) and Enterprise Service Bus (ESB) can be effectively conducted by a virtual team of service brokers. Currently, VEI is accomplished through SOA and ESB using web services and business process engines that execute WSDL and WS-BPEL. To reengineer a Virtual Enterprise (VE) based on one or more legacy components, referred to as a legacy VE, the abstract and concrete parts of the relevant business processes of the VE need to be reverse engineered to a high level of abstraction. To develop new business processes, business process requirements need to be forward engineered into business processes in BPEL. However, service brokers need guidelines for comprehending the operations of the legacy VEs or for understanding the business process requirement. In order to provide clear communication of this information, we propose a UML model driven Business Process Development Methodology (BPDM) called mBPDM. We demonstrate its applicability and capabilities by applying it to two case studies: a loan application process system which involves reverse engineering and Washington State Patrol's Drug Recognition Evaluation system which involves forward engineering. Based upon the results of reverse and forward engineering of two virtual enterprise cases, the guidelines, which use UML as a blueprint with multi-architectural views, help service brokers understand the underlying process architecture and organization of a virtual enterprise that has been built using the SOA concept and the contemporary ESB.**

Keywords-Virtual Enterprise Integration; Business Process Development Methodology; Enterprise Service Bus

I. INTRODUCTION

We define a virtual enterprise (VE) as a temporary organization of cooperating agents that unites to achieve a common purpose by sharing knowledge, information, resources, and processes. The sharing, cooperation, and coordination are supported by computer network infrastructure. The common purpose typically involves faster adaptation to changing requirements, conditions, or opportunities; effective use of resources, or dynamic reallocation of processes. According to [9], a virtual enterprise is "based upon the ability to create temporary co-operations and to realize the value of a short opportunity that partners cannot (or can, but only to lesser extent) capture on their own." It needs the support of extensive use of information and communication technologies [16]. The advent of Service-Oriented Architecture (SOA) allowed information technology professionals to implement the core resources of enterprises as services and connect the services across the enterprises on computer networks.

In SOA [22], the software developer uses services as fundamental building blocks to their software development process. Standard service interface language such as Web Service Description Language (WSDL) and interaction protocol such as Service-Oriented Architecture Protocol (SOAP) are popular implementation technologies for SOA. Many research results in [2, 19, 21, 23, 24] are shown how SOA using SOAP web services has affected the integration of virtual enterprise.

However, the traditional role of a developer is clearly classified into three loosely coupled roles in SOA: service consumer/requestor, producer/provider, and broker. One of a service broker's tasks is to compose available web services to create new applications to solve complex business problems on top of SOA's native capabilities [13].

The need for creating new business processes composed of multiple web services to solve complex business problems is behind the development of Enterprise Service Bus (ESB). An ESB is "a standard-based integration platform that combines messaging, web services, data transformation, and intelligent routing to reliably connect and coordinate the interaction of significant numbers of diverse applications across extended enterprises with transactional integrity [3].

Currently, an ESB is implemented through both web services and business process engines that execute WSDL and Web Services Business Process Execution language (WS-BPEL). WS-BPEL is an emerging industry standard language of how web services are arranged and composed to solve complex business problems and has the full backing of industry leaders such as Microsoft [12], Oracle [15], Sun Systems [20], IBM [8], as well as the open source community

978-1-4244-5336-8/09/$26.00 ©2009 IEEE

[14]. In fact, Microsoft, Oracle, and IBM as well as many other companies and open source projects are vying for positions in this new technology and actively developing tools to utilize it [17]. Many of them are at their infant stages[1] for automatic and dynamic composition.

Although the SOA architectural pattern and the ESB integration platform allow service brokers to develop integrated virtual enterprises, service brokers must overcome several challenges in order to do this effectively: 1) a service broker need reverse engineering guidelines in order to comprehend and model the legacy business processes implementing the existing virtual enterprise integration. 2) A service broker needs forward engineering guidelines in order to transform business process requirements into BPEL process models.

In this paper, we propose a UML model driven Business Process Development Methodology (BPDM) called mBPDM that will address these needs. We apply mBPDM to two case studies: a loan application process system for reverse engineering, which was developed by Oracle, and Washington State Patrol's Drug Recognition Evaluation (DRE) System for forward engineering. Based upon the results of reverse and forward engineering of two virtual enterprise cases, the guidelines help a service broker to conduct how a virtual enterprise is built by using the SOA concept and the contemporary ESB.

II. RELATED WORKS – BUSINESS PROCESS DEVELOPMENT METHODOLOGY (BPDM)

In [7], Fowler mentioned that UML models can be used in three different ways for modeling a software-intensive system: 1) UML as a sketch, 2) UML as a blueprint, and 3) UML as a programming language.

In order for a service broker to fully make use of these SOA and the benefits of ESB, there needs to be solid methodologies that will guide the broker to develop executable business processes on an ESB and comprehend the business processes already implemented in BPEL. If business processes requirements are documented and can be shared between service brokers for BPEL implementation, more user-satisfied business processes can be implemented. If a BPEL implementation can be re-documented in a higher level abstraction, the legacy business processes can be easily modernized based upon new requirements or business environment changes. Therefore, we are interested in UML as a blueprint of BPEL for both forward and reverse engineering. Currently, there are a few business process development methodologies for the forward engineering of given business process requirements into composite services that are executed on an ESB. In addition, no methodology exists for reverse engineering a composite service to business requirements that can be understood by a service broker.

[1] Among them, Oracle's Business Process Engine was mature to be tested at the time of writing this paper and was employed in this paper.

Singh and Hunhns provide a methodology for describing a business process in terms of activity and class diagrams in [18]. For example, BPEL process definition is represented as a class with a stereotype <<process>>. BPEL activities such as 'receive', 'reply', and 'invoke' are represented as <<receive>>, <<reply>>, and <<invoke>> activities. They only show how BPEL can be visualized in UML instead of vendor specific graphic notations. From this perspective, they use UML as a sketch of a business process.

Mantell proposed UML2BPEL toolkit that transforms a given business process in UML to its corresponding BPEL and WSDL files in [11]. Activity and class diagrams are used to represent a business process. Neither reverse engineering from BPEL to UML nor management of business process was requirements discussed. Mantell uses UML as a programming language.

Erl [6] proposed the SOA delivery. The gap between business logic and application logic is fulfilled by service interface layer, which consists of other three layers: application service, business process, and orchestration layers. The SOA delivery lifecycle is applied to the development of service interface layer. The lifecycle consists of six phases: service-oriented analysis, service-oriented design, service development, service testing, service deployment, and service administration. These phases are similar to those used for object-oriented development projects except for introducing unique consideration in every phase of service construction and delivery. Each phase consists of several steps. For example, the service-oriented analysis phase consists of three steps: 1) define analysis scope, 2) identify automation systems, and 3) model candidate services.

Although Erl's SOA delivery lifecycle provides good high level general guidelines for people who build the service interface layer, neither modeling technique nor reverse engineering of existing virtual enterprises using the service interface layer were discussed. No UML documentation was used.

Service-Oriented Software Reengineering (SOSR) methodology, which is close to our approach, was proposed by Chung et al [5] to support software developers to reengineer a legacy software system into a service-oriented software system(s) based on SOA. The SOSR methodology is based upon service orientation, architectures (Model-View-Controller (MVC), 3-layered, n-tier, SOA), roles and its specific tasks using RACI (Responsible, Accountable, Consulted, and Informed) charts, and UML (Unified Modeling Language) modeling. However, this methodology needs to be extended for use by software developers to accommodate composite web services based application development. Although they provide the foundation for SOSR and propose key concepts in the area of composite web services, they mentioned that future work is required for examining how SOSR relates to this issue.

III. MODEL-DRIVEN BUSINESS PROCESS DEVELOPMENT METHODOLOGY

One of unique characteristics of an ESB is that it allows business and IT to build a closer relationship. This is due to the fact that orchestration of web services represents higher-level abstraction called business process by hiding traditional middleware objects that have been used to support business-to-business interactions [1]. Furthermore, business needs can be directly translated into business process applications through composite web services.

The model-driven Business Process Development Methodology (mBPDM) guides a service broker to visualize, specify, document, and construct business processes in terms of reverse or forward processes, sub-roles of a service broker over time, Kruchten's 4+1 views, and UML diagrams. Chikofsky and Cross [4] provide a framework for examining reengineering technologies by defining important terms and identifying common objectives of them such as forward and reverse engineering. Reengineering starts with reverse engineering phase that re-documents a given legacy system in a high level abstraction to comprehend the legacy system. Through the re-documentation process, a software engineer recovers designs that were embedded into the system. After comprehension, the forward engineering phase is conducted to either restructure or modernize the legacy system to a new target system. These concepts can be applied to reverse or forward engineering of a virtual enterprise. Figure 1 shows reverse and forward mBPDMs.

FIGURE 1: REVERSE AND FORWARD MODEL-DRIVEN BPDMS

Since the reverse process is the opposite of forward process, the reverse process is only described in this paper due to the limited paper size. At the reverse process, a service broke takes a business process in BPEL and re-documents it in a high-level abstraction, i.e. a UML model. The process consists of deployment, implementation, static design, dynamic design, and analysis phases in sequence. A sub-role of the service broker is changed at each phase, which is shown in Table 1.

TABLE 1: SUB-ROLES AND INTERESTS AT EACH BUSINESS PROCESS (BP) PHASE

Phase	Sub-Role	Main Interests
BP Deployment	BP Administrator	• ESB
BP Implementation	Service-Oriented Programmer	• SOP
Static BP Design	BP designer	• Structure of messages and services
Dynamic BP Design	BP designer	• Workflow of a BP
BP Analysis	BP analyst	• BP requirements

At the deployment phase, the service broker is interested in deploying and administering business processes on an ESB. The service broker is interested in the implementation of a business process in BPEL at the implementation phase, which we call Service-Oriented Programming (SOP), compared to traditional Object-Oriented Programming (OOP). At the static design phase, the classes and their relationships of a business process and common external representations in XML are interests of the service broker. At the dynamic design, the service broker is interested in how service methods of partner services are invoked and what their messages in XML are exchanged. Finally, the service broker describes use-cases and can comprehend what business requirements were actually implemented in the BPEL process at the business process analysis phase.

Each sub-role of the service broker at a specific phase brings different views on the business process development. The mBPDM heavily uses the 4+1 view model of software development architecture of Kruchten [10]: The architecture of software-intensive systems needs to be understood in term of multiple concurrent views to address the concerns of various stakeholders separately in terms of the functional and non-functional requirements. The description of software architecture is organized around four views: logical (or static design) view, implementation view, process (or dynamic design) view, deployment view, and then illustrated by use cases, or scenarios that becomes the fifth view. The "4+1" view model of a business process in Figure 2 allows the service brokers to find what they want to know about the business processes in terms of abstract/executable and static/dynamic perspectives while focusing on business process requirements.

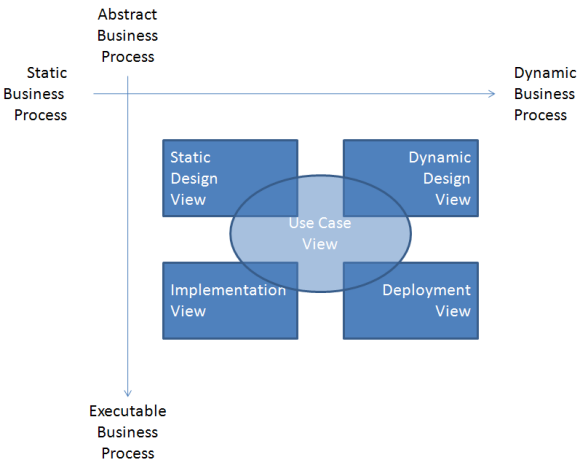

FIGURE 2: 4+1 VIEWS AND BUSINESS PROCESS

Based upon this 4+1 view concept, Table 2 shows which UML diagrams of UML 2.0[2] are used for each view. There are total 13 diagrams. Among 6 structure diagrams, class, component, package, and deployment diagrams are used. Use case and activity diagrams are used among 3 behavior diagrams. Sequence diagram is used among 4 interaction diagrams.

TABLE 2: THE RELATIONSHIPS BETWEEN 4+1 VIEWS AND UML DIAGRAMS

View	UML Diagram
Use Case View	• A use case diagram for uses cases and their actors such as a service consumer, service brokers, and service producers
Static Design View	• A class diagram for web services and their methods • A class diagram for XML message scheme
Dynamic Design View	• A sequence diagram for message exchanges between web services • An activity diagram for workflow
Implementation View	• A component diagram for composite services and their partner web services with their URLs
Deployment View	• A deployment diagram for nodes and their networks of ESB with service components

First of all, the deployment view shows the nodes that form the system's hardware topology on which the system executes. This view takes into account the distribution, delivery, and installation of the parts that make up the physical system. The existing system is analyzed as it is and presented in a deployment diagram. In mBPDM, the deployment view shows the nodes of service producers (or partnerLinks in the BPEL terms) and service brokers and their connections with the SOAP protocol. The '.wsdl' components of partnerLinks and processes and the '.bpel' components of the business processes are shown as executable artifacts.

Secondly, one looks at the implementation view that encompasses the components and files that are used to assemble and release the physical system. This view primarily addresses the configuration management of the system's releases, made up of loosely coupled components and files that can be assembled in various ways to make a running system. In mBPDM, the implementation view shows at where the WSDL components of partnerLinks and the BPEL components and their corresponding WSDL components are located. Each URL of a service producer or broker is represented as a package. The dependency relationships between components are represented. For example, a process's WSDL component depends upon its corresponding BPEL component. The BPEL component depends upon its partner WSDL components.

Thirdly, one looks at the static design view that includes the classes and interfaces that make up the system. This view primarily supports the functional requirements of the system, meaning the services that the system should provide to its end users. In mBPDM, the design view shows a static description of business process classes. A BPEL process is considered as a business process class. Its variables are considered as the attributes of the class. The operation of the client partnerLink is considered as an operator of the business process class.

Fourthly, the process or dynamic design view takes into account the dynamic behaviors of objects, i.e. message exchanges among objects. In mBPDM, either sequence or activity diagram are employed. The sequence diagram shows message exchanges between the business process class and its partnerLink classes. The activity diagram shows the workflow of the business process.

Lastly, one looks at the use case view that encompasses the use case scenarios that describe the behavior of the system as seen by its end users, analysts, and testers. This view doesn't specify the organization of a software system; rather, it exists to specify the forces that shape the system's architecture. In mBPDM, business process requirements are described and managed. It will help any service broker to easily understand what business processes need to be designed and implemented for VEI.

IV. REVERSE ENGINEERING OF A LOAN APPLICATION PROCESS SYSTEM IN BPEL

A loan application process system was developed by Oracle to demonstrate Oracle's Business Process Engine called Oracle BPEL Process Manager [15]. The loan application process system invokes a composite service that consists of three services. The reverse mBPDM is used for a service broker to comprehend the composite, i.e. a business process in BPEL.

The loan application process consists of a sequential business process that takes a loan application document as

[2] Introduction to OMG's UML http://www.omg.org/gettingstarted/what_is_uml.htm

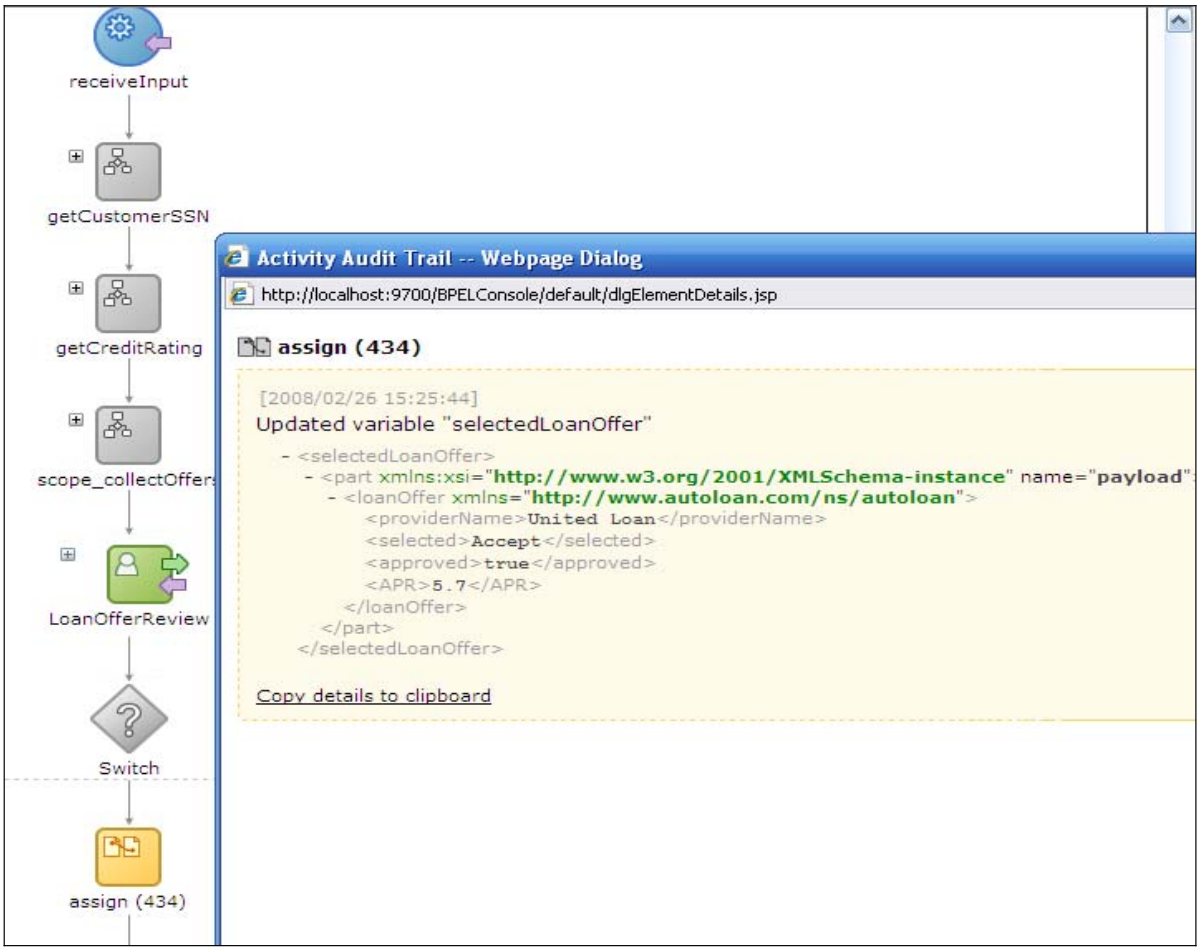

FIGURE 3: A BUSINESS PROCESS IN BPEL CALLED 'LOANFLOWPLUS' FOR THE LOAN APPLICATION PROCESS

input then it returns a selected approved loan offer as the final output. First of all, the user enters personal information on a webpage to start the loan application process. Next, the information is sent to a composite service called 'LoanFlowPlus'. The composite service consists of three simple services: 'CustomeServiceDesign,' 'CreditRating Service,' and 'LoanService.'

The 'CustomeServiceDesign' returns the Social Security Number (SSN) of the applicant from an Enterprise Java Bean (EJB) (accessed as a service through the native EJB Web Services Invocation Framework (WSIF) binding). The SSN is then used to retrieve the credit rating of the applicant by calling a synchronous 'CreditRatingService' web service to request a credit rating for that customer. Once the credit rating is obtained, it sends the completed loan application to a loan processing service called 'LoanService' that finishes that application process and provides an interest rate offer and delivers it to the customer. The selected and approved offer is returned as a result of the flow. The customer can then accept or reject the offer from the process. The automation of the application process saves time and energy as well as reduces the amount of human error that can occur if done manually. Figure 3 shows how the composite service in BPEL that consists of three services is designed for the loan application process by using Oracle Business Process Manager.

The service broker's role is sub-classified into administrator, programmer, designer, and analyst. The first step is to understand the current deployment of the application as a system administrator. A deployment diagram represents the deployment: The 'LoanFlowPlus' composite service is composed of three simple services that are provided by three service producers, respectively.

Once the system's current deployment is analyzed, then the implementation view is created for a (Service-Oriented) programmer followed by the design view for a designer, the process view for a developer, and the use case view for an analyst, respectively.

V. ENGINEERING OF DRUG RECOGNITION EVALUATION SYSTEM

In this section, the forward BPDM is applied to the Washington State Patrol's Drug Recognition Evaluation (DRE) System. The Washington State Patrol department currently has a Microsoft Access based Windows form application that keeps track of drug evaluations performed by DRE officers. Drug evaluations are performed on drivers that are suspected of driving under the influence of drugs. A series of tests as well as personal information are documented on a hardcopy of a DRE form. The hard copy form is then sent to an office administrator of the Impaired Driving Section. Also, a blood sample is sent to the toxicology lab with the suspect's personal information. After the lab tests are performed at the toxicology lab, the hardcopy results are sent by mail to the office administrator of the Impaired Driving Section. The office administrator then enters all the information into the DRE MS Access application. This total process take at least three months and sometimes an indefinite amount of time as the paper trail gets broken and information lost. This inhibits data and information to be reported in a timely, accurate and usable fashion. This process needs to be automated and is perfect for a forward engineering project using mBPDM to be undertaken.

The DRE application exists on a shared drive and is unstable with limited functionality. Significant reports cannot be generated and accessed over the Internet. This application is being reengineered using ASP .NET and VB .NET technologies. The .NET technologies provide a stable and high performance environment that allows more data to be processed and a significant amount more information to be reported. These reports are available over the web to many different parties. This greatly enhances the ability of the patrol department in its fight against impaired driving.

This project is chosen in this research because there are many similar applications that exist in work places around the globe. With the advent of MS Access, an explosion of standalone desktop applications has occurred. As business needs change and growth occurs, many of these applications need to be reengineered to web based distributed systems that are performance driven and highly scalable.

User case studies reveal that the first area of input comes from the DRE officer who conducts the evaluation on the suspect and enters in the initial information in a form that generates a unique rolling log number. A DRE officer means an officer from around the State and many different law enforcement agencies, as well as the Washington State Patrol. The second area of input is from the State Toxicology Laboratory that provides the Drug results. The drug results of the blood tests will be accessible through a web service using the rolling log number. The final area of input will be the data entry of the rest of the form that will take place in Olympia or Seattle, Washington State.

Web services will be implemented to allow the data entered in by the field DRE officers and the toxicology results from the State Toxicology Lab to be inserted into the final form. This process is ripe for BPEL to orchestrate the web services into a cohesive process.

This paper focuses on the service broker for BPEL composite services. The service broker generates a model describing each of 4+1 views. The first view developed in the model is the use case view. Figure 4 shows the use case view that is created in terms of the broker using the guidelines created in the extended SOSR for the DRE application. An analyst employs the use-case view to discover services from published services that could be used for the new application. In terms of the DRE application, the analyst discovered that there was a web service under development by the toxicology lab that published toxicology results as a web service. The input for this web service was a rolling log number and the output was the drug result for the suspect.

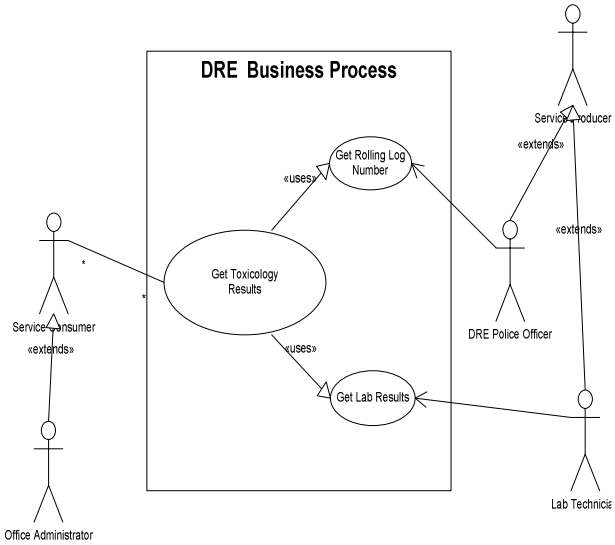

FIGURE 4: DRE APPLICATION USE CASE VIEW OF SERVICE BROKER ROLE

Secondly, the process view for the broker is created. A designer employs the process view to describe discovered services and to compose a composite service using available services. For the DRE application, one can see in this view that the service consumer initiates the process by calling the BPEL DRE process. The DRE process then calls the 'RollingLogService' and the 'ToxWebService' respectively. These need to be constructed in this order as the unique license number generates the rolling log number that in turn is used by the 'ToxWebService' to return the drug result. The BPEL process returns back to the consumer the toxicology result for a subject suspected of driving under the influence of drugs. This process is shown in Figure 5.

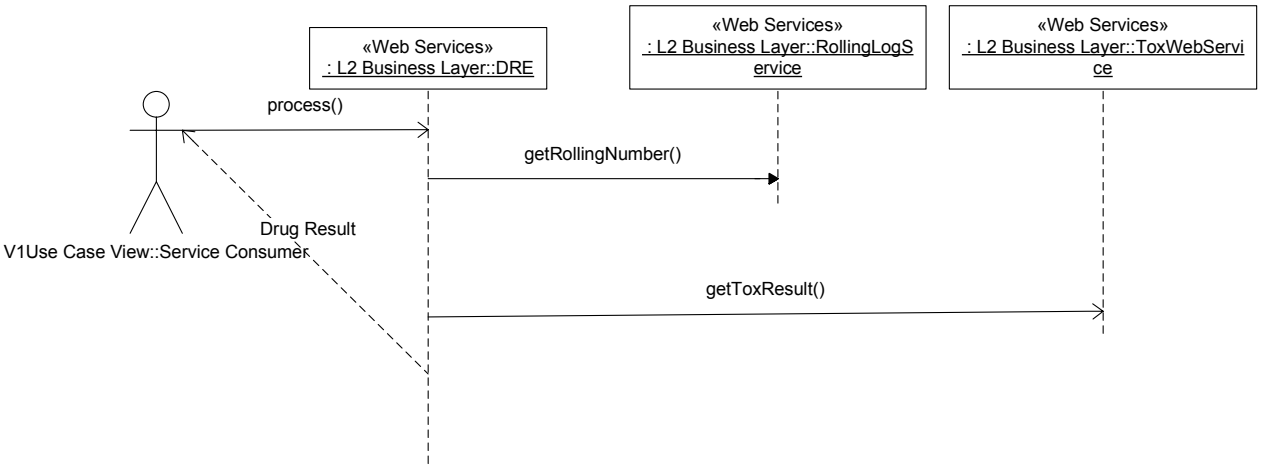

FIGURE 5: DRE APPLICATION PROCESS VIEW OF SERVICE BROKER ROLE

Thirdly, the design view is created for the broker. A designer employs the design view to describe services and design composite services. Figure 6 shows the design view that was created for the DRE application. The design view shows that one business process needs to be created for the BPEL composite service. The process method will start the BPEL process. The 'RollingLogService' has one method called 'getRollingNumber' that will retrieve a rolling log number given a unique license number. This web service called 'RollingLogService' will need to be designed and implemented as it was discovered in the use case view stage by the analyst that there weren't any web services available to accomplish this task. On the other hand, the analyst was able to determine that there was a service that could be used to return Toxicology results from the Toxicology Lab. According to the application documentation the 'ToxWebService' has one method called 'getToxResults.' This method returns toxicology results returned from the 'getRollingNumber' method. The input and output for this web service will need to be integrated into the BPEL composite service.

Fourthly, the implementation view is created for the broker. A BPEL programmer uses this view to implement the application. Figure 7 shows the implementation view created for the DRE application. The programmer uses the implementation view to implement the program using the BPEL designer. This view shows the process that needs to be created and the web services that need to be called in the BPEL composite service. The process that is created is called the 'DRE.bpel.' The BPEL Programmer uses Oracle JDeveloper to design the BPEL composite service and call the appropriate web services in the correct order in accordance with the business process shown in the implementation view.

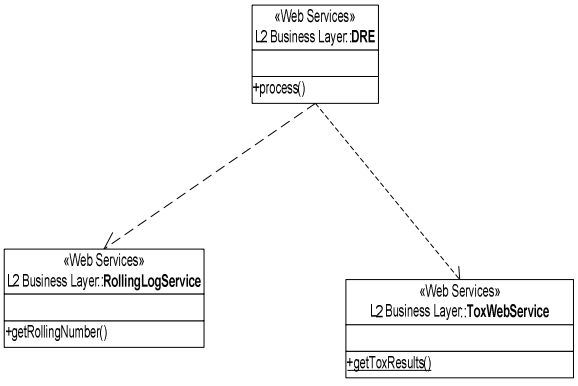

FIGURE 6: DRE APPLICATION DESIGN VIEW OF SERVICE BROKER ROLE

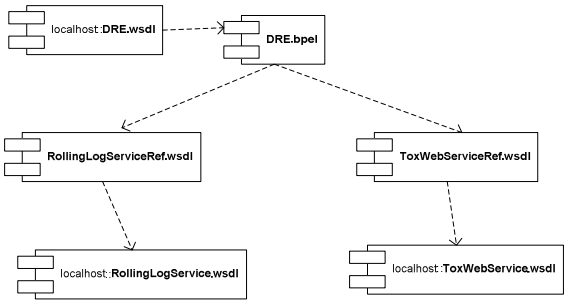

FIGURE 7: DRE APPLICATION IMPLEMENTATION VIEW OF SERVICE BROKER ROLE

Finally, the deployment view is created. A system administrator uses the deployment view to deploy the composite service. The DRE composite service is deployed on the Oracle BPEL PM Server. The 'RollingLogService' and the 'ToxWebService' are also deployed on the 'localhost.'

A successful DRE application was developed in accordance with the extended SOSR and a BPEL process was implemented to orchestrate this process. This process eliminates the bottlenecks and the human errors that occur when hardcopies of documents are transcribed by humans and sent by mail. The automated process saves both time and money with results that can be updated virtually instantaneously.

In the final BPEL composite service, first a user enters a unique license number. The process then calls the first web service to retrieve the unique rolling log number for this license number. Then this rolling log number is used to retrieve the drug results from the toxicology lab via web services. Finally, the toxicology drug result is returned to the BPEL process for that particular subject and populates the evaluation form that will be finished by an office administrator in the Impaired Driving Section of the Washington State Patrol.

VI. CONCLUSIONS AND FUTURE RESEARCH

Based upon the results of reverse and forward engineering of the two virtual enterprise cases, the model-driven business process development methodology, mBPDM, enables a service broker to understand how to build a virtual enterprise by using the SOA concept and the contemporary ESB. We first show how a service broker can comprehend the current virtual enterprise that was built based upon SOA and ESB. A reverse model-driven BPDM is proposed and applied to a legacy virtual enterprise's re-documentation, the visual model, which shows the static/dynamic and logical/physical views of a legacy virtual enterprise.

Secondly, we demonstrate how a service broker can compose web services for business logic and deploy composite services onto an ESB. A forward model-driven BPDM is proposed and applied to business logic modeling. The business logic is visualized, specified, documented and constructed. A service broker then implements the business logic in BPEL and deploys it as a new service onto an ESB.

Based on our evaluation of the results, the mBPDM needs to be further researched and improved. This methodology needs to be applied to more complex and diverse composite services and the outcomes need to be discussed.

REFERENCES

[1] F. Casati, E. Shan, U. Dayal, and Shan, M. "Business-oriented management of Web services," Communications of ACM, Vol. 46, No. 10, 2003, pp. 55-60.

[2] C. Chunlai and F. Jingchun, "Application Integration of Virtual Enterprises in the South-to-North Water Division Program," Proceedings of 2008 International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII08), Vol. 2, 2008, pp.345-348.

[3] D. E. Chappell, Enterprise Service Bus, O'Reilly Media, Inc. 2004.

[4] E. J. Chikofsky3and H. J, Cross II. "Reverse Engineering and Design Recovery: A Taxonomy," IEEE Software Vol. 7, No. 1. Jan. 1990. pp. 13-17.

[5] S. Chung, S. Davalos, J. An, and K. Iwahara. "Legacy to Web Migration: Service-Oriented Software Reengineering (SoSR) Methodology," International Journal of Services Sciences 2008, Vol. 1, No.3/4, pp. 333 – 365.

[6] T. Erl, Service-Oriented Architecture Concepts, Technology, and Design, Prentice Hall, 2005.

[7] M. Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition), Addison-Wesley Object Technology Series, 2004.

[8] IBM, Retrieved on March 4, 2008, http://www.ibm.com/developerworks/library/ specification/ws-bpel/

[9] B. R. Katzy and G. Schuh, "The virtual enterprise," Handbook of Life Cycle Engineering: Concepts, Methods and Tools, New York, Chapman & Hall, 1997.

[10] P. B. Kruchten, "The 4+1 View Model of Architecture," IEEE Software, 12 (6), 1995, pp. 42 – 50.

[11] K. Mantell, From UML to BPEL, 2005, http://www.ibm.com/developerworks/library/ws-uml2bpel/?ca=dnt-436

[12] Microsoft Biztalk, Retrieved on March 4, 2008, http://www.microsoft.com/biztalk/default.mspx

[13] N. Milanovic and M. Malek, "Current Solutions for Web Service Composition," IEEE Internet Computing, Nov.-Dec. 2004, Volume: 8, Issue: 6, pp. 51-59.

[14] Active Endpoints, Open Source Project, Retrieved on March 4, 2008, http://www.activevos.com/community-open-source.php

[15] Oracle. Process Manager, Retrieved on March 4, 2008, http://www.oracle.com/technology/products/ias/bpel/index.html

[16] H. K. Park and J. Farrel, J. Virtual enterprise — Information system and networking solution, Computer and Industrial Engineering, Vol. 37, Issue 1-2, 1999, pp. 441-444.

[17] Scientific Research Corporation, "2007 Comparative Research Methodology Process For BPEL," http://www.active-endpoints.com/documents/documents/1/2007-comparative-assessment-for-BPEL.pdf

[18] M. P. Singh and M. N. Huhns, Service-Oriented Computing semantics, Processs, Agents. John Wiley & Sons, Ltd, 2005.

[19] F. Sun, L. Wang, T. Chen, and Y. Qu, "Dynamic Information Integration of Virtual Enterprises Based on Web Services and J2EE," Proceedings of 2007 International Conference on Wireless Communications, Networking and Mobile Computing (WiCom 2007), Sept. 21-25, 2007, pp. 6146-6149.

[20] Sun Microsystems, Retrieved on March 4, 2008, http://developers.sun.com/jsenterprise/nb_enterprise_pack/reference/techart/bpel.html

[21] W. Tan, J. Xue, and J. Wang, "A Service-Oriented Virtual Enterprise Architecture and its Applications in Chinese Tobacco Industrial Sector," Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE06), 2006, pp. 95-101.

[22] S. Weerawarana, F. Vurbera, F. Leymann, T. Dtorey, and D. Ferguson, Web Services Platform Architectures. Upper Saddle River, NJ: Prentice Hall, 2005.

[23] Y. Xu, J. Shen, and Z. Chen, "Ontology-based Information Retrieval of Web Services in Virtual Enterprise," Proceedings of 2004 IEEE International Conference on Services Computing (SCC'04), 2004, pp.441-444.

[24] B. Zhou, Z. He, J. Tang, "An adaptive model of virtual enterprise based on dynamic web service composition," Proceedings of the Fifth International Conference on Computer and Information Technology (CIT'05), 2005, pp.284-289