Toward Next Generation Distributed Business Information Systems: Five Inherent Capabilities of Service-Oriented Computing

Chung, Sam and Davalos, Sergio

Abstract— The research conducted examines how the emerging Service Oriented Computing (SOC) paradigm enables the development of Distributed Business Information Systems (DBIS) and thus influences the acceptance of DBIS. For this purpose, a Service Oriented Architecture (SOA) that can be implemented by currently available SOC technologies is used. The SOA is applied to the development of DBISs for retail Supply Chain Management (SCM) and stock currency conversion. To analyze the use of the SOC paradigm on the development of DBISs, the following five inherent capabilities of SOC are proposed: interoperability, reusability, adaptability, interdependability, and agility. We investigate how the proposed capabilities are naturally provided by the SOC paradigm in the development of the DBISs. These capabilities enable clearer communication between software and business professionals by allowing 'software as a service' and 'business process as a service.'

Index Terms—Adaptability, Agility, Business-to-Business Application Integration, Distributed Information System, Enterprise Application Integration, Interdependability, Interoperability, Reusability, Service Oriented Architecture, Service Oriented Computing, Supply Chain Management

1. INTRODUCTION

The purpose of this research is to examine how the Service-Oriented Computing (SOC) paradigm enables the development of Distributed Business Information Systems (DBIS and thus influences the acceptance of DBIS through its five natural capabilities - interoperability, reusability, adaptability, interdependability, and agility. Web services, a promisingly better technology for integrating heterogeneous applications, has recently emerged [1, 2, 3].

A web service uses standard interface definitions and general interaction protocols based upon eXtensible Markup Language (XML).

Manuscript received September 26, 2005. This work was supported in part by the University of Washington Tacoma under Grant Founder's Endowment.

S. Chung is with the Computing & Software Systems program, Institute of Technology, University of Washington Tacoma, USA (e-mail: chungsa@u.washington.edu).

S. Davalos is with the Milgard School of Business, University of Washington Tacoma, USA (e-mail: sergiod@u.washington.edu).

For the integration of applications, software developers have adopted the SOC paradigm in the belief that the SOC paradigm can overcome implementation complexity involved with the current distributed object computing paradigms [1, 2].

In this research, two models of Service Oriented Architecture (SOA) are described that are based on currently available SOC technologies: 'publish-discovery-binding' and 'publish-compose -discovery-binding' models. The SOA models are then applied in the development of DBISs for retail Business Information System (BIS), Supply Chain Management (SCM) [4], and stock currency conversion.

For the retail BIS, two heterogeneous BISs are developed to support the operations of a retailer and a supplier. The business logic layer for each BIS is implemented with web services. Web services are published to public service registries. The two BISs are then integrated with the published web services for Business-to-Business (B2B) transactions between the retailer and the supplier.

The primary purpose of the second application, a Stock Currency Converter (SCC) system, is to examine the use of a composite web service on a business process execution engine and utilizing Service-Oriented Programming (SOP). A composite web service is built by composing two existing web services.

The use of SOC on the development of DBISs is examined based upon the proposed five natural capabilities of the SOC paradigm. In terms of the five capabilities, naturally provided by the SOC, the case study results show that the SOC paradigm is very capable for developing Distributed BISs requiring Enterprise Application Integration (EAI) and B2B Application Integration (B2BAI). Both software and business professionals share the common conceptual building block – a service: software as a service and business process as a service.

This paper consists of six sections. In the next section, two different types of application integration based on the SOA and its application to EAI and/or B2BAI are discussed. In Section 3,

we propose and define five inherent capabilities of SOC based upon the SOA that can be used for evaluating the development of DBIS. In section 4, two case studies are presented that require successful use of the five capabilities in a DBIS environment. One case study involves B2B transactions in retail SCM, which need EAI and B2BAI. The second case study continues the examination of a DBIS where Stock Currency Converter (SCC) system transactions are implemented by using .NET web services and SOP. In Section 5, we discuss the results of the application of the SOC paradigm to the development of distributed BISs in terms of the five capabilities. In Section 6, we conclude that the five inherent capabilities show why the SOC computing paradigm is very effective for EAI as well as B2BAI. Also, we show how it brings about a significant programming paradigm shift from object orientation to service orientation so that the gap between software and business professionals can be significantly reduced.

2. BUSINESS APPLICATION INTEGRATION AND SOA

In this section, two different types of application integration are discussed: EAI and B2BAI. Then, we investigate the SOA that can be supported by current commercial web service platforms. We also introduce examples of web services to real business applications that need both EAI and B2BAI, such as retail SCM.

Depending on where integration happens, there are two different types of application integration: EAI and B2BAI [1]. Although applications in an enterprise generally tend to use the same language and platform, it is quite possible to have to contend with heterogeneous platforms and languages for various applications for EAI. Since applications belong to the same organization, explicit application interfaces and the semantics of business transactions between applications can be implicitly assumed. However, in B2BAI, this assumption does not hold. Standardized interfaces and general protocols for application and system interactions need to be explicitly specified and agreed upon by the different stakeholders.

Web services, an emerging technology, can be used for both EAI and B2BAI since the technology supports the development and use of standard interfaces for integration and general protocols for interactions among heterogeneous applications. Web services are self-describing, open components that support rapid, low-cost composition of distributed applications. Web services can communicate with other web services through a variety of Internet protocols and XML formats [5, 6, 7]. Web services are based on open standards: Simple Object Access Protocol (SOAP), Web Service Definition Language (WSDL), and Universal Description, Discovery and Integration (UDDI).

SOA is a conceptual architecture for organizing and understanding the development and use of web services. Based on the roles of participants, there are two types of SOA: 'publish-discovery-binding' and 'publish-compose-discovery-binding' model. Figure 1 shows an SOA using the "publish-discovery-binding" model. A web service provider implements web services for addressing business logics. WSDL is used to describe standard interfaces of the available services. The web services are "published" to a web service registry – a UDDI. A web service registry contains information such as the service location, the provider, etc. A service broker (SB) uses UDDI specifications to define the registry service for web services. A service requestor (SR) or consumer (SC) can "discover" web services from the registry. Based upon the discovered service information, the SC's application can be "bound" to the services at execution time through SOAP. SOAP is used to transfer the message between heterogeneous applications. SOAP is a lightweight XML-based messaging protocol that is used for requesting/responding to messages over a network using existing transport protocols such as HyperText Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), or Multipurpose Internet Mail Extensions (MIME). The HTTP protocol is most popular as the transport protocol of SOAP.

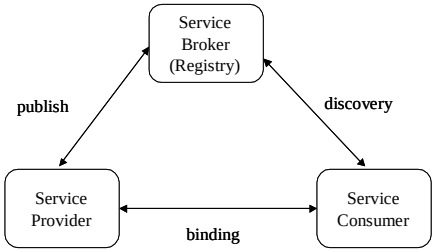

Figure 1. A SOA using the 'publish-discovery-binding' model

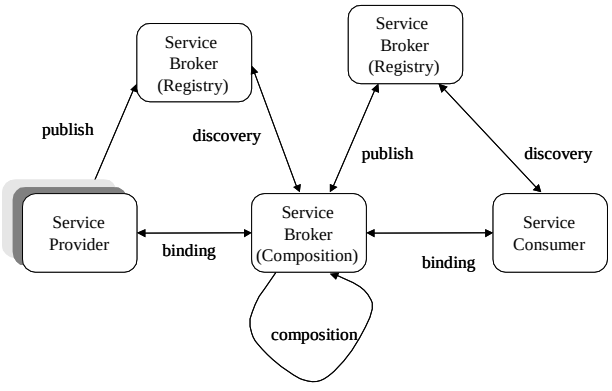

Figure 2. A SOA using the 'publish- compose-discovery-binding' model

A set of composite services called a business process can be represented in Business Process Execution Language (BPEL). A composite

service in BPEL is executed on a business process execution engine such as Oracle BPEL engine [8]. Figure 2 shows a SOA using the 'publish-compose-discovery-binding' model. A service consumer discovers a required service and invokes the discovered service. A service provider develops a service in a programming language and publishes its service specification, UDDI specification, to a service registry. A service broker composes a set of services as a new service and administers the registry. Since protocols and specifications of the current SOA are still evolving, new SOAs may be proposed in the near future.

3. FIVE NATURAL CAPABILITIES OF SOC

The inherent capabilities of the SOC paradigm are useful for analyzing the effects of the SOC paradigm on the development of Distributed BISs. Due to the heterogeneity of platforms, networks, and languages, the necessary capability is interoperability. Also, in terms of software engineering, the reusability capability has been emphasized for reduction in the cost of software maintenance and evolution. In addition, there are adaptability, interdependability, and agility. Although the current SOC related technologies need to be enhanced, we argue that the following five capabilities are naturally provided for distributed information systems by the SOC paradigm using the SOA: interoperability, reusability, adaptability, interdependability, and agility.

Interoperability is "the ability of information systems to operate in conjunction with each other encompassing communication protocols, hardware software, application, and data compatibility layers [9]." Interoperability is the key property of any distributed computing paradigm where there is heterogeneity between the implemented distributed systems. Since software systems are typically developed using different platforms and computer languages, a distributed computing paradigm that can integrate applications of heterogeneous software systems is needed. The SOC paradigm, using the SOA, has this capability by providing a standard interaction protocol SOAP, a standard interface description WSDL, and a standard XML based external data representation.

Reusability is defined as "the ability to design software modules in a manner so that they can be used again and again in different systems without significant modification [10]." Reusability enables software developers to reduce the development time by reusing existing code. In developing distributed information systems, software developers have even attempted to reuse remote methods. Reusability is inherent to the SOC paradigm using the SOA since the SOC paradigm is an evolution of a distributed computing paradigm that supports reusability through object orientation concepts. In addition, more reusability is gained through collaboration and/or composition of existing web services.

Adaptability is "the ability to change or be changed to fit changed circumstances [11]." Adaptability enables software developers to design new distributed systems with minimal effort when circumstances change. Adaptability is inherent to the SOC paradigm using the SOA since the SOC paradigm provides service related architectural and design patterns and service registries for the developers. By understanding the service patterns before developing a new service based distributed system, the developers can adjust their architecture and designs. Also, since specification of web services can be published to a UDDI registry and software developers can access the registry, software developers can adjust the selection of web services for their new projects based on current demands and requirements.

We define interdependability as the capability to minimize coupling and maximize cohesion. "Coupling is the level of interdependency between a method and the environment, (other methods, objects and classes), while cohesion is the level of uniformity of the method goal [12]." The level of interdependability is very useful for team collaboration since it is important to know who will take which roles among the service requestor, broker, and provider. A web service requestor implements only the interface and web service invocation. If a user interface needs a business function, the business function is invoked as a method in the form of a web service. A web service provider implements the service component. The web service is a (object-oriented) component with an interface. A web service broker handles the advertisement of the web service. Since role specific views for service requestor, broker, and provider are clearly defined in the development of a service-oriented distributed system, interdependability is inherent to the SOC paradigm using the basic SOA.

Agility is "the ability to respond quickly to short-term changes in demand or supply and handle external disruptions smoothly [13]." Agility enables an organization to respond quickly to demands. Agility is inherent to the SOC paradigm using the "compose-publish-discovery-binding" model since the SOC paradigm enables service composition for the developers. Instead of developing new software components, software developers can compose existing web services and invoke the composite web service to respond the demands. A web service composition can be model-driven with lightweight coding, which we call Service-Oriented Programming (SOP). SOP can enable software developers to deliver product releases in much shorter periods of time compared to the traditional Object-Oriented Programming (OOP).

4. *CASE STUDIES*

4.1 SERVICE-ORIENTED RETAIL SCM SYSTEM

A Retail SCM was developed as a set of business processes that coordinates and integrates the flow of goods from supplier to retailer to customer in order to reduce inventory and expedite prompt and precise services [4]. However, a BIS in a retail store is usually an isolated application and/or in most cases tightly coupled with its main office's information systems. Moreover, the information system cannot easily exchange data with information systems of other suppliers.

There is considerable interest in the development of web-based SCM applications. The Web Services Interoperability (WS-I) organization used SCM as a B2B example [14, 15]. WS-I has links on SCM architecture and use case models. Alonso et al. used SCM examples to explain web services coordination protocol and composition [1]. In addition, there are examples of applications of web services to real business problems. Schmerken describes the application of SOA to B2B application integration in Finance [16]. Maresca et al [17] applied SOA to the design of the Cargo Community System for the transport operators located in the Port of Genoa. Goepfert and Whalen describe three case studies of web service application in real business such as access management module, provisioning module, sales force automation module, chat rooms, message boards, instant messenger, web polls, etc [18].

In order to analyze how the emerging SOC paradigm will influence the development of DBISs, the transactions between business information systems of retailers and suppliers are first considered. For purpose of simplification, only one retailer and one supplier are considered in this example. The number of participants can be increased if necessary. The transactions are described with the following four use cases:

- Use Case 1 - Retailer's purchase order entry**:** The retailer's purchasing department enters the purchasing order that will be stored into the retailer's database. The retailer sends the purchasing order form to the supplier's sales department.
- Use Case 2 - Supplier's invoice entry: The supplier's sales department issues an invoice statement based upon the retailer's purchase order and its shipping information. That invoice statement is then sent to the retailer's inventory department.
- Use Case 3 - Supplier's purchasing order retrieval from retailer: The supplier's sales department looks over the purchase order and processes the order. Then the supplier's shipping department fulfills the order and updates the purchasing order to show the number of items shipped.
- Use Case 4 - Retailer's invoice retrieval from supplier: The retailer looks over the invoice to confirm the quantity ordered, quantity shipped and the correct prices. Then, the invoice can be closed. If not, then contact the supplier's with the discrepancies.

The SOA is used for implementing the use cases. The main advantage of using the SOA is the fact that the current commercial web services platforms such as Microsoft .NET or IBM Websphere support it. In the SOA, a synchronous interaction protocol (SOAP/HTTP) is used between a service requestor and a service provider. For B2BAI, web services are published onto an external public web service registry that can be accessed by any organizations that participate in business transactions. To exchange data, a domain specific XML format is employed. (In this paper, IxRetail is used for data interchange [19].) Since we can assume that a service consumer knows where web services are located for EAI, web services for EAI are not published onto a service registry, contrary to the web services for B2BAI.

Based on the four use cases, the following web services are implemented:

- The RetailerWS class provided by the retailer for EAI: The retailer's web client can communicate with the retailer's database and load a list of suppliers from the retailer's database.
- The SupplierWS class provided by the supplier for EAI: The supplier can retrieve the list of Retailers and purchase order from the supplier's database. The supplier checks available new purchase order through a web interface. The web interface will upload a list of retailers from the supplier's database using a private web service called SupplierWS.
- The RetailerDB2IxRetailWS class provided by the retailer for B2BAI and published to a public service registry: The supplier can select the retailer and retrieve a list of purchase order dates from the retailer's database through a method from the public web service published by the retailer called RetailerDB2IxRetailWS.
- The SupplierDB2IxRetailWS class provided by the supplier for B2BAI and published to a public service registry: The retailer can pull the available invoice dates from the supplier's database.

Table 1 shows the use case employed in this case study, web services implemented, and the service requestors or consumers. In the use case 1, a user interface is created for the data entry in order for a retailer to create purchase orders. The purchase order system was created with Microsoft Visual Studio .NET as a Windows client application using C# as the programming language due to the heavy interaction between

the user and the application.

Table 1. The Information of Web Services

Use Case	Web Service Class Name	Web Service Requestor
1	RetailerWS	Retailer
2	SupplierWS	Supplier
3	SupplierWS RetailerDB2IxRetail WS	Supplier
4	RetailerWS SupplierDB2IxRetail WS.	Retailer

Since the purchase order application is an internal process, it uses a private web service called RetailerWS to communicate with the retailer's database. The RetailerWS is a private web service because the web service is published onto a public UDDI registry. The RetailerWS class contains methods that will add a new supplier, retrieve all supplier IDs, delete a supplier, retrieve a supplier's information, and save the purchase order information.

In the use case 2, once the shipment has been fulfilled, the supplier's sales department has to enter the quantity shipped information into the Order Fulfillment Form. This part of the supplier process uses private web services. It uses the SupplierWS class to retrieve the list of retailers and a purchase order from the supplier's database. It selects the purchase order that has already been inserted into the supplier's database. The shipped quantity information is entered into the supplier's database through the Order Fulfillment Form. The web client application retrieves the purchase order from the database and the Order Fulfillment form to update the units shipped.

The use cases 3 and 4 will discuss that the BISs interact with each other to process B2B transactions. In order to use a web service of the other BIS, the client application has to search for it from a known public web registry. For this project, the web services are published onto the Microsoft's UDDI web service registry.

For example, in the use case 4, a retailer retrieves the invoice from the supplier's database. The retailer's web application client can upload a list of suppliers from the retailer's database through the RetailerWS class. Then it pulls the available invoice dates from the supplier's database through the public web service called SupplierDB2IxRetailWS. Once the retailer selects the invoice, it will retrieve the invoice from the supplier's database and convert it into IxRetail format by a method in the SupplierDB2IxRetail WS web service. At the completion of that function, an XML file will be returned to the retailer. The process will call another web service IxRetail2RetailerDBConverterWS to convert that XML file into the retailer's database structure.

4.2 STOCK CURRENCY CONVERTER

In the Stock Currency Converter (SCC), there are investors from foreign countries who get the stock quote of a company within the United States. The SCC web application receives inputs for both a country name and a stock symbol to be quoted, and displays the quoted stock prices in the selected country's monetary unit.

Service providers publish software components to a service registry as Web Services, exposing their interface so that a service consumer can access services exposed by the service provider. A service registry, in most cases, will be present to aid the service consumer in discovering services published by the service providers. The interactions between SC, SB, and SP are modeled by using a Domain Specific Language (DSL) [20, 21, 22] or Unified Modeling Language (UML) that can be converted into codes.

The service consumer is first interested in the type of user interface. Based upon the user requirement, a web application is selected for 24*7 service regardless of location. Also, the service consumer is interested in discovering and invoking necessary services from a service broker(s). The discovery can be done manually or programmatically, with or without intelligence, or implicitly/explicitly. Since the discovery is done manually and implicitly in this case, there is no interaction between service consumer and service broker for the discovery. The assumption is that the service consumer knows what services are available and where they are located at in advance.

An executable model, compared to expressive visual models in UML, can represent the diagram by using Microsoft Visual Studio 2005: The web application 'SCCWebApplication' invokes a web service called 'CurrencyStock'. Its diagram is described in Microsoft Domain Specific Language (DSL) and generates an executable framework in C# and ASP .NET.

The service broker is interested in service discovery, matchmaking, and composition. By using a BPEL composition method [23], the service broker composes a new composite service by using two services 'getRate' and 'getQuote'. The service broker discovered two web services 'Currency Exchange Rate' and 'Delayed Stock Quote' through the 'xmethods.com' service registry. This composition is model-driven, which is shown in Figure 3. By using Oracle's BPEL designer and BPEL as the domain specific language for service composition modeling, two services are composed into a new service which is published as a new web service.

Service-Oriented Programming is applied here because we are using the concept of SOA to integrate software components in order to create a new software component. In the stock quote

currency conversion example, there are service providers called 'xmethods.com' that first publish services to a services directory. Then, we discover services that are useful to us and call those services directly, and the results are combined to form a new result useful to a user. This new application using BPEL becomes another service that others might invoke directly or be composed with another service(s).

The interactions between all the participating partners are described using the syntax of BPEL, and then deployed and executed by the BEPL engine. The Oracle BPEL Designer and Process Manager are tools that provide users with a development environment to create, deploy, and execute applications using BPEL. The BPEL Designer provides a visual modeling approach for developers to create new applications. This approach is a form of Model Driven Architecture, where a system is designed with visual techniques and a code will be generated according to the visual diagrams. Here the visual diagrams are converted to BPEL syntax. A deployment tool is also provided through the BPEL Designer. The end product will then be executed by the BPEL engine, which will serve BPEL services. All of the tools and techniques to help build Service Oriented Programs will be part of the software factory.

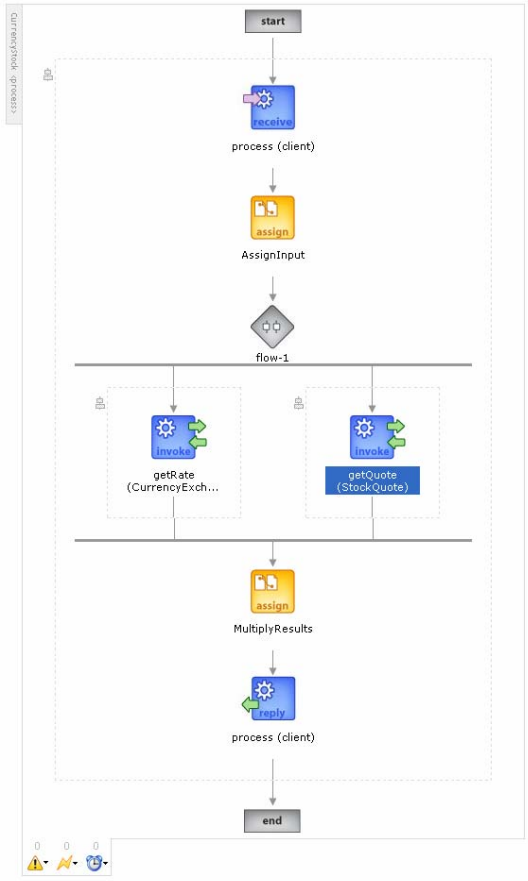

Figure 3. A Model-Driven Service Composition using Oracle Domain Specific Language for a Service Broker

5. *ANALYZING THE FIVE INHERENT CAPABILITIES*

Based upon the case study results, we examine how the SOC paradigm provides the five inherent capabilities: interoperability, reusability, adaptability, interdependability, and agility, which are necessary for successful development of distributed, enterprise-level BISs.

First of all, this Case Study 1 demonstrates that the SOC paradigm provides interoperability for integrating heterogeneous applications. For example, the use case 3 in Table 1 employs two different types of web service classes: SupplierWS and RetailerDB2IxRetailWS. The SupplierWS class was provided by the supplier and used for internal enterprise integration. The RetailerDB2IxRetailWS class was provided by the retailer and used for integrating the supplier's and the retailer's applications. Therefore, the interoperability allows the software engineers to attempt to integrate heterogeneous systems for either EAI or B2BAI.

Secondly, this Case Study 1 shows that the SOC paradigm provides reusability for software developers. In Table 1, the use case 3 employs the SupplierWS class that was implemented by the supplier for the use case 2. The use case 4 uses the RetailerWS class that was implemented by the retailer for the use case 1. Therefore, the reusability supports the software developers to reuse existing web services.

Thirdly, Case Study 1 shows that the SOC paradigm provides (minimal) adaptability for software developers. Since the architectural pattern SOA in Figure 1 is also used for implementing BISs of both the retailer and the supplier, a developer can adjust the SOA for the supplier's BIS if the developer understands the architecture of the retailer's BIS. In addition, since specifications of web services for B2BAI are published to a UDDI registry in Figure 2, software developers can access the registry and select web services for their new projects easily. The use of web service registries encourages software developers to adapt their distributed systems to the changed circumstances by allowing the discovery of existing web services. Therefore, adaptability encourages the software developers to use the architectural pattern and existing services for developing a new BIS seamlessly. If the semantics of services are provided and automatic discovery of services on demand is fully provided in the near future, the adaptability can be significantly improved.

Fourthly, Case Study 1 shows that the SOC paradigm provides interdependability for software developers. There are two different types of user interface: windows client application and web client application. Both of them use the same web service class RetailerWS. If the retailer requests a web service of the supplier, the retailer does not have to know the

implementation of the service. The retailer can only focus on his or her own role as a service requestor. If there are any changes to the implementation of the business logic and its service description remains constant, the service broker is independent upon the changes in terms of providing the services of web service discovery. Therefore, the interdependability of service requestors, brokers, and providers brings more clear roles to a software development team. It also brings a more precise boundary between presentation logic through user interfaces and business logic through web services.

Lastly, Case Study 2, shown in Figure 3, depicts that the SOC paradigm provides agility for software developers. Since the existing services can be easily composed according to the given demand and its model-driven service-oriented programming enables the service broker to develop his or her service with minimal programming efforts, the service broker is able to respond quickly to short-term changes in demand. If the automatic composition of services at execution time is fully provided in the near future, the agility can be significantly improved.

6. *CONCLUSIONS*

Although the current SOC related technologies need to be enhanced and are being evolved, we argue that the current SOC paradigm using the SOA is suitable for developing distributed business information systems because the following five capabilities are naturally provided: interoperability, reusability, adaptability, interdependability, and agility. Since this SOC paradigm provides these inherent capabilities for software developers, the SOC paradigm is very effective for not only EAI within an organization but also for B2BAI between organizations. Although other distributed object-oriented computing paradigms such as CORBA or RMI may support interoperability and reusability, they cannot easily obtain the other three capabilities such as adaptability, interdependability, and agility. The employed case studies demonstrate that those features can be supported by the premature SOC technologies currently used. The common conceptual building block with the five inherent capabilities, service, will enhance the communication between software and business professionals and proliferate the concepts of software as a service and business process as a service amongst both professional communities.

REFERENCES

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

[2] Zimmermann, O., Tomlinson, M., and Peuser, S. Perspectives on Web Services: Applying SOAP, WSDL, and UDDI to Real-World Projects. *Springer Verlag*. 2004.

[3] Ferris, C. and Farrell, J. What are Web services? *Communications of the ACM*. Vol. 46, No. 6. pp. 31. June 2003.

[4] Kumar, K. Technology for Supporting Supply Chain Management. *Communications of the ACM*. Vol. 44, No. 6. pp. 58-61. June 2001

[5] Turner, M., Budgen, D., and Brereton, P. Turning Software into a Service. *IEEE Computer*. Vol. 36, No. 10. pp. 38-44. October 2003.

[6] Chung, J., Lin, K., and Mathieu, R. Web Services Computing: Advancing Software Interoperability. *IEEE Computer*. Vol. 36, No. 10. pp. 35-37. October 2003.

[7] Papazoglou, M., and Georgakopoulos, D. Service-Oriented Computing. *Communication of the ACM*. Vol. 46, No. 10. pp. 25-28. October 2003.

[8] Oracle (2005). BPEL Process Manager. [Online] Available: http://www.oracle.com/technology/products/ias/bpel/index.html

[9] Definitions of Interoperability (2005) [Online] Available: http://www.ichnet.org/glossary.htm

[10] Definition of Reusability. (2005) [Online] Available: http://myphliputil.pearsoncmg.com/student/bp_hoffer_modernsad_3/glossary.html

[11] Definition of Adaptability (2005) [Online] Available: http://www.cogsci.princeton.edu/cgi-bin/webwn

[12] Definition of Coupling and Cohesion (2005) [Online] Available: http://www.ugolandini.net/AccoppiamentoCoesioneE.html

[13] Agility. From Wikipedia (2005) [Online] Available: http://en.wikipedia.org/wiki/Agility

[14] Sample Application Supply Chain Management Architecture. {2005) [Online] Available: http://www.wsi.org/SampleApplications/SupplyChainManagement/2002-11/SCMArchitecture-0-11-WGD.pdf

[15] Supply Chain Management Use Case Model (2005) [Online] Available: http://www.wsi.org/SampleApplications/SupplyChainManagement/2002-11/SCMUseCases-0.18-WGD.pdf

[16] Schmerken, I. (2005, September) Can Web Services Live Up to the Hype? *Wall Street & Technology Online*. [Online] Available: http://www.wallstreetandtech.com/story/mag/showArticle.jhtml?articleID=14702913&pgno=1

[17] Maresca, M., Baglietto, P., and Zingirian, N. (2002) Deployment Experience of Service Oriented Architecture in the Business Community of the Port of Genoa: Lessons Learned. *XML Europe 2002 – Conference*. Barcelona, Spain. [Online] Available: http://www.idealliance.org/papers/xmle02/dx_xmle02/html/abstract/02-03-03.html

[18] Goepfert, J. and Whalen, M (2002, August) An Evolutionary View of Software as a Service. *IDC*. [Online] Available: http://itpapers.zdnet.com/abstract.aspx?&tag=tu.web.ont.dir1&scid=90&docid=46783

[19] *The Association for Retail Technology Standards (ARTS) of the National Retail Federation*. [Online] Available: http://www.nrf-arts.org

[20] Greenfield J. and Short, K. Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools. *Object Oriented Programming Systems Languages and Applications (OOPSLA) 2003*. Anaheim, CA. pp. 16-27. October 26-30, 2003

[21] Greenfield, J., Short, K., Cook, S., and Kent, S. Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. *Wiley Publishing, Inc*. Indianapolis, IN. 2004.

[22] Domain-specific programming language from Wikipedia. (2005). [Online] Available: http://en.wikipedia.org/wiki/Domain-specific_programming_language.

[23] Milanovic, N., and Malek, M. Current Solutions for Web Service Composition. *IEEE Internet Computing*, Vol. 16. pp. 51-59. November/December 2004.