Modeling Service-Oriented Software Development: Services Ecosystem

Sam Chung
Computing & Software Systems
Institute of Technology
University of Washington, Tacoma
1600 Commerce St.,
Tacoma, WA 98402
1-253-692-5866
chungsa@u.washington.edu

Abstract - The purpose of this paper is to propose a novel modeling approach called Services Ecosystem that applies the concept of ecosystems in ecology to Service-Oriented Software Development and Integration. For this purpose, an ecological system for software systems is proposed for the emerging Service-Oriented Computing paradigm, describing how participants interact with each other within their environments. Three emerging concepts, Service-Oriented Programming, Software Factories, and Service Grid, are employed to explain biotic and abiotic environments. Based upon the Services Ecosystem model, we demonstrate Services Ecosystem Model transformations by using a case example. The Services Ecosystem model is a novel approach for envisioning the Service-Oriented Computing paradigm in terms of an ecosystem in which the roles/perspectives of each participant and their relationships/interactions to environments are clearly described with a holistic view.

Keywords: Service-Oriented Architecture, Web Services, Model-Driven Development

1 Introduction

Service-Oriented Computing (SOC) has emerged as a software development paradigm that can minimize the differences in the perspectives of business and software professionals. The original intent of SOC was to support the development of software systems through the use of standardized software components that are declared as services and can be interoperable within heterogeneous software systems [1, 2]. The SOC paradigm enables the design of software systems as a set of services, i.e. Software as a Service (SAAS) [3]: Software systems can thus be recursively constructed as a set of services that employ standardized service interfaces and interaction protocols.

However, modern software developments using the SOC paradigm require many participants to be involved with different perspectives, such as publishing, discovering, composing, invoking services, etc. Those participants interact with one other and cope with the constantly changing requirements and evolving computing environments. The different perspectives of participants and interactions between them and their environments require modeling with a holistic view. In this paper, we propose a novel approach to modeling the participants and their interaction with a Services Ecosystem that applies the concept of 'biological ecosystem' (or ecological system) to software development. The concept of ecosystem model has been introduced to explain how organic community behavior occurs in response to environment in ecology and environmental science [4]. Based upon the ecological hierarchy for software systems and the ecosystem model of Odum and Barrett [4], a Services Ecosystem model is proposed for depicting interactions between service consumers and producers on user requirements within programming, product development, and infrastructure environments. These environments are described by using Service-Oriented Programming (SOP), Software Factories (SF), and Service Grid (SG) concepts, respectively. Based upon the Services Ecosystem Model, we demonstrate and discuss Services Ecosystem model transformations with an example using web services and a composition technique.

2 Related Works

Various attempts have been made to provide conceptual models of SOA (Service-Oriented Architecture). A popular approach is to explain the SOA in terms of

technology stack models such as SOAP, WSDL, BPEL OWL-S, UDDI, etc [5]. Since the technology stack model views the SOA in terms of available languages and protocols, the model needs to be changed with emerging technologies. While the technology stack model is very useful for software professionals, but is not so for business professionals.

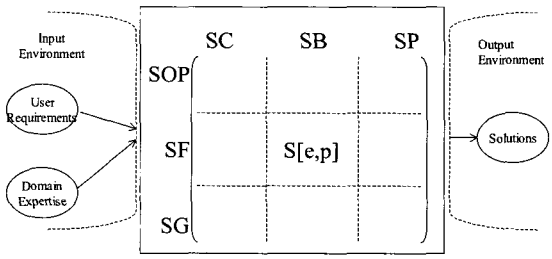
Based on the three service stakeholders (service consumer, publisher, and provider) and their interactions, the basic SOA model has been broadly accepted in the SOC community [1,6]. The basic SOA model assumes that the first-generation of web services technologies, exemplified by WSDL, SOAP, and UDDI, is used. A business process conceptual building block that can be understood and used by business professionals has not been introduced yet in this basic SOA model. Papazoglou proposed an extension to SOA [6] from the basic SOA to include the second-generation web services technologies such as service orchestration, service transaction management, etc. Three different types of services - basic, composite, and managed services - were introduced for several stakeholders such as service client, service aggregator, service provider, service operator, and market maker. However, the extended SOA does not explicitly show how business processes are designed and executed in terms of SOA stakeholders on the service grid infrastructure.

Zimmermann et al described a hybrid Service-Oriented Analysis and Design (SOAD) modeling approach for SOA projects that combines the elements of OOAD, Enterprise Architecture (EA) frameworks, and Business Processing Modeling (BPM) [7]. Although this approach is unique in terms of integrating existing modeling approaches for SOA projects, these approaches were not mentioned in terms of interactions between participants and their environments.

3 Services Ecosystem

In order to emphasize the interactions of project participants and their environments, the term "ecosystem" has been used in several articles without referring to an ecosystem model in ecology [8, 9, 10]. In this paper, the ecosystem model of Odum and Barrett in [4] is used. In [4], Odum and Barrett define an ecosystem as a community with biotic (living) and abiotic (nonliving) environment. Also, an ecosystem is a black box in its input and output environments. The ecosystem model of Odum and Barrett is based on energy, consumers, producer, storage, and biotic and abiotic environments.

In this research, user requirements and domain expertise are defined as energy sources in the input environment. Also, the solutions, which may be delivered by software applications, are considered as the components of the output environment. Then, we define an ecosystem in terms of a matrix, which we call the Services Ecosystem Matrix. The Services Ecosystem Matrix S [e, p] is defined for a type of environment e ∈ E and a type of participant p ∈ P. There are interactions between two adjacent elements. In the SOC paradigm, E is the set of three different types of environments: programming, development, and execution. We introduced three different environments such as Service-Oriented Programming (SOP), Software Factories (SF), and Service Grid (SG). P is the set of participants such as Service Consumer (SC), Service Broker (SB), and Service Provider (SP). This model is shown in Figure 1.



S [e, p], where e ∈ E ={SOP, SF, SG} and p ∈ P={SC, SB, SP}

Figure 1. A Services Ecosystem

A SC discovers a required service and invokes the discovered service either directly or through a service broker. A SP develops a service in a programming language and publishes its service specification to a service registry. A service broker composes a set of services as a new service and administers the registry.

Service providers publish software components to a service registry as web services, publicizing 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 using a Domain Specific Language (DSL) [11] or Unified Modeling Language (UML) that can be converted into code.

At the next level, we have a software factory that utilizes Service Oriented Programming. Jack Greenfield and Keith Short proposed the software factory concept in [11]. It has been applied to the development of Microsoft Visual Studio 2005. At this level, we provide patterns, DSLs, and schemas that build templates targeted at various domains.

At the bottom level resides a Service Grid (SG) environment. We define a service grid as a set of distributed service nodes that are interconnected through an Enterprise Service Bus (ESB) [12], which is shown in Figure 2. A service node is a service platform that can play a role in requesting, brokering (discovering, composing,

and executing), managing, and publishing a service. A set of service nodes on the service grid is integrated on demand to build a large-scale, distributed, heterogeneous, and trustworthy software system at either design or even execution time. The ESB maintains the protocols that make up the Service Oriented Architecture such as SOAP, HTTP, and WS-* related protocols. Also, network protocols and the physical network connection are all part of this layer. This layer also takes care of service routing, load balancing and resource management facilities.

The SCC Services Ecosystem matrix in Figure 2 shows the three participants interact with each other in the three given internal environments. Each element S [e, p] has its own process or methodology. Three possible approaches are possible to build the matrix: environment-first, participant-first, and environment-participant first Services Ecosystem modeling. The environment-first approach focuses on the interactions of all participants within a specific environment. The participant-first approach focuses on the interactions of a participant within all three environments. The environment-participant approach focuses on the interactions of a participant with a specific environment. Then, the modeling process is expanded to adjacent environment-participant elements until all environment-participant elements are modeled.

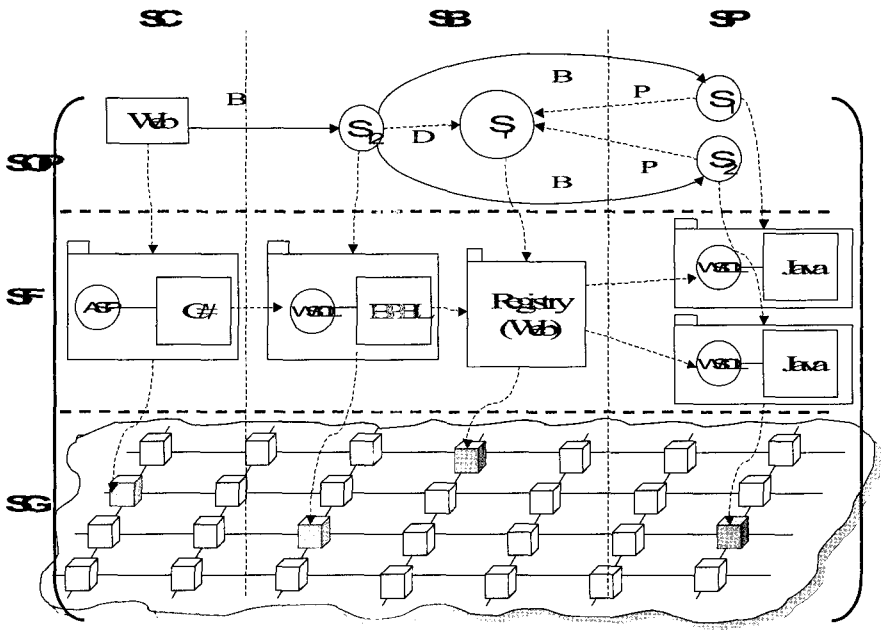


Figure 2. A Concern Matrix of the SCC Services Ecosystem

4 Case Example

Any service-oriented software development can be modeled and developed by using its own Services Ecosystem. Let's consider the following case example: In the Stock Currency Converter (SCC), there are investors who are from foreign countries that want to get the stock quote of a company within the United States. An investor needs to get the stock quote in US Dollars and convert the quote on their own using a stock trading company ABC's web site. 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.

The environment-participant approach is chosen to build the matrix of the SCC services ecosystem in Figure 2 For example, let's select S [e, p], where e = SOP and p = SC. Among nine mappings, only two mappings are chose for explanation:

- S [SOP, SC] - Mapping of Web Application to the Service-Oriented Programming Environment by Service Consumer.
- S [SOP, SB] - Mapping of BPEL to the Service-Oriented Programming Environment by Service Broker

S [SOP, SC]: 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. The object of this service is shown in the S [SOP, SC] element of Figure 2. Also, the SC is interested in discovering and invoking necessary services from a service broker(s). The discovery can be done manually/programmatically, with/without intelligence, or implicitly/explicitly. Since the discovery is done manually and implicitly in this case, there is no interaction between SC and SB for the discovery. The assumption is that the SC knows what services are available and where they are located at in advance. The SC invokes the service S12. The SC does not know and does not need to know the design and implementation of the service S12 since S [SOP, SC] and S [SOP, SB] are loosely coupled through web services. The interaction with label B (Binding) exists between S [SOP, SC] and S [SOP, SB] in Figure 2 that is similar to a UML activity diagram.

An executable model, compared to expressive visual models in UML, can represent the diagram in Figure 3. The web application 'SCCWebApplication' invokes a web service called 'CurrencyStock'. This diagram is described in Microsoft domain specific language and generates an executable framework in C#.

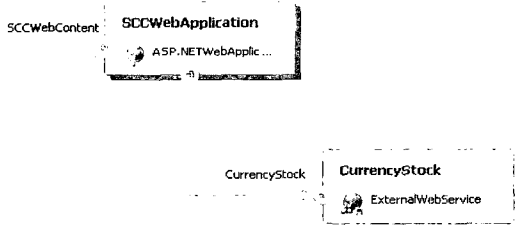


Figure 3. A Model-Driven Development using Microsoft DSL for S [SOP, p] where p∈{SC, SB}

[SOP, SB]: The SB is interested in service discovery, matchmaking, and composition. By using a BPEL composition method [13], the service broker composes a new composite service S12 by using two services S1 and S2, shown in Figure 2. The SB 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 4. 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 service-oriented architecture to integrate software components in order to create a new software component. With the stock quote currency conversion example, there were service providers that first published services to a services directory such as Xmethods.com. Then, we discovered the services that were useful to us and invoked those services directly, and the results were combined to form a new result useful to a user. This new application created using BPEL will also become another service that others might invoke directly or to compose it with another service. It will be published in the service registry. Many instances of SOA occur as we create applications using the SOP approach.

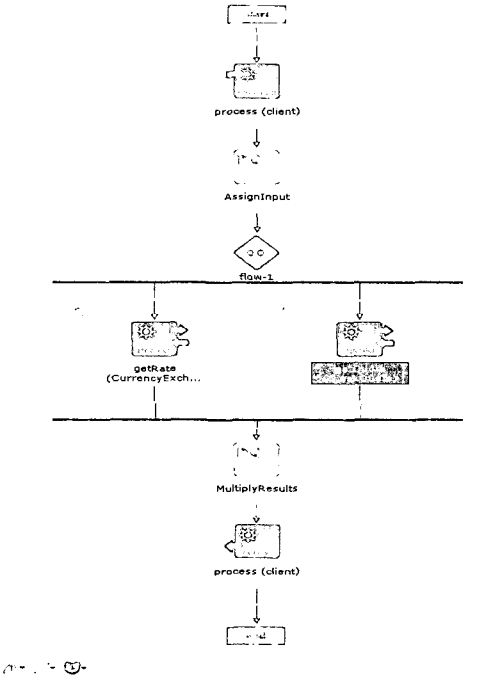


Figure 4. A Model-Driven Service Composition using Oracle Domain Specific Language for S [SOP, SB]

By repeating this environment-participant approach for all 9 elements, the functional diagram of the SCC Services Ecosystem model in Figure 2 is built. Also, we summarize the required technologies for each component of an environment for a specific participant in Table 1.

5 Analysis

From both the concern matrix in Figure 2 and the technology matrix in Table 1 for the SCC Services Ecosystem, our analyses show that the service ecosystem model brings multiple and coordinated views to all participants in a service-oriented software development and integration from the least fine-grained view to the most coarse-grained view for all participants and their interactions across all environments.

Table 1. A Technology Matrix for the SCC Services Ecosystem

Env.	Component	Consumer	Broker	Provider
SOP	MDD	SOMDD (Distributed System Designer with MS DSL)	SOMDD (Oracle BPEL Designer with Oracle DSL)	? (OOMDD)
	SOA	Discovery, binding	Discovery, composing, publishing, binding	Publishing
	Languages	MS DSL, XHTML, ASP .NET	Oracle DSL	? (UML Class Diagram)
SF	MDD	OOMDD (Visual Programming for Web Form, Class Designer with MS DSL)	SOMDD (BPEL)	? (OOMDD)
	Languages	C# .NET	BPEL, WSDL	Java, WSDL
	Framework	.NET Framework 2.0		Java Class Hierarchy
	Pattern	MVC, Layered Architecture, Multi-Tier Architecture, C/S Architecture	Layered Architecture Multi-Tier Architecture	? (CBD, Layered Architecture, Multi-Tier Architecture)
	Process	XP, FDD	Composition method – BPEL XP, FDD	? (XP, FDD)
	Tools	MS Visual Studio 2005 Beta (Distributed System Designer, Class Diagram Designer), Web Browser	Oracle BPEL Designer	? (Java SDK, Java2WSDL, ECLIPSE)
SG	Service Node	VM (.NET CLR), SOAP Engine (MS IIS)	BPEL Engine (Oracle BPEL Engine)	SOAP Engine (WebMethods Glue)
	ESB	SOAP, HTTP	SOAP	SOAP

The Services Ecosystem matrix S is defined for each environment $e \in E$ and each role of a participant $p \in P$, where E is the set of three different types of environment for service-oriented programming, development, and execution and P the set of participants for service consumer, service broker, and service provider. First of all, the holistic view on a software project is provided through the Services Ecosystem Matrix, S [e, p], where $e \in E=\{SOP, SF, SG\}$ and $p \in P = \{SC, SB, SP\}$. The software project manager needs to know who is working in which part of the project under which development environments and infrastructure.

In addition to the holistic view of the Services Ecosystem Matrix S, which is the most coarse-grained view for all participants, the least fine-grained view for a participant in a specific environment is also provided in terms of an element of S. An element of S [e, p] is a fine-grained view that describes which participant needs to be involved in which environment. Its interactions with adjacent elements describe how the view can be transformed to other views.

6 Conclusions

The results of this research show that modern software developments using the SOC paradigm can be modeled by using a biological ecosystem concept. Modern software development is very similar to an ecosystem in nature since various participants with different views for a software project interact with each other and their environments.

There are several key benefits of this approach. Firstly, since the Services Ecosystem model provides a holistic view of service-oriented software development to all participants considering their service-oriented programming, development, and execution, the efforts of all participants can be easily orchestrated. Secondly, since a specific view of each participant within the same environment shows the interactions of participants within the environment, the boundaries of activities of each participant are clearly defined.

This Services Ecosystem model is the first step toward understanding service-oriented software development and integration in terms of interactions among project participants and their project environments. Other ecology-based concepts that have been researched in ecology need to be studied and transformed into the environment of the Services Ecosystem model. For example, natural selection, speciation, and evolution can be studied for business process adaptation, composition, and evolution. Currently, we are studying the representation, composition, and adaptation of business process collaboration in terms of attributes and traits of organisms and population change over time, based upon the Services Ecosystem model.

References

[1] Thomas Erl. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice Hall. 2004.

[2] Steffen Staab. (2003). Web Services: Been There, Done That? *IEEE Intelligent Sys*tems. January/February 2003. Vol. 18, No. 1. pp. 72-85.

[3] Mark Turner, David Budgen, and Pearl Brereton. Turning Software into a Service. *IEEE Co*mputer. October 2003. Vol. 36, No. 10. pp. 38-44.

[4] Eugene P. Odum, Gary W. Barrett. *Fundamentals of Ecology*. Thomson Brooks/Cole, 5th Ed, Belmont, CA, 2005.

[5] Frank Leymann. (2003). *Web Services: Distributed Applications without Limits*. Invited Talk and Joint Opening Speech at BTW 2003 and KiVS 2003. Proceedings Database Systems For Business, Technology and Web BTW 2003.

[6] Mike P. Papazoglou. *Service -Oriented Computing: Concepts, Characteristics and Directions*. Proceedings of the Fourth International Conference on Web Information Systems Engineering 2003. December 10 - 12, 2003. pp. 3.

[7] Olaf Zimmermann, Pal Krogdahl, and Clive Gee. *Elements of Service-Oriented Analysis and Design: An interdisciplinary modeling approach for SOA projects*. June 2004.

[8] PC Magazine. The Web Services Ecosystem. Available at http://www.pcmag.com/image_popup/0,1871,s=1479&iid=28020,00.asp

[9] IDC. European Application Services Ecosystems. Available at http://www.idc.com/getdoc.jsp?containerId=IDC_P7193

[10] Jim Highsmith. *Agile Software Development Ecosystems*. Addison Wesley Professional. 2002.

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

[12] David A. Chappell. *Enterprise Service Bus*. O'Reilly. 2004.

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