Web Services Software Architecture

Syahrul Fahmy

School of Informatics, The University of Manchester,
PO Box 88, Manchester M60 1QD, United Kingdom
S.Abdul-wahab@postgrad.manchester.ac.uk

Abstract. Web services have received widespread attention and acceptance in the software engineering community. Automatic composition of services, to meet user's requirements, is a powerful mechanism to ensure satisfaction of varied needs and thus enable the vision of Web services. Although Web services are equipped with the fundamental concepts and supporting technologies, the architectural style of the composite service composed from Web services is fixed by the prescriptions of the *Service Oriented Architecture*, and does not provide the level of flexibility stipulated by alternative service-based approaches such as the *Service-Based Software* vision. To enable the composition of Web services using alternative architectural styles, the author proposes that software architecture be included as a parameter in the composition process. This is enabled by the use of the Alfa framework to model the desired architectural style, and the use of this information to support service discovery.

1 Introduction

Web services are configurable software services that use open standards and protocols to connect and integrate distributed components for creating and managing computer applications. Web services have been becoming a major software trend because they provide means for integrating data sources and data transfer between applications. Web services share business logic, data and processes through a programmatic interface across a network. In order for web services to be operational, interoperability of applications over intra/internet is necessary. This interoperability requires a technology that is independent of hardware, operating system, and software specific technologies. As such, specific standards are used to address these interoperability issues that consist of XML to define structured data, SOAP as a messaging protocol, WSDL to define interfaces, and UDDI as a registry to publish available services.

There are two approaches an organization can adopt to take advantage of Web services namely the *Service Oriented Architecture* (SOA) and the *Service-Based Software* (SBS) approaches. In the SOA approach, a fixed architecture is presumed for service composition. Little flexibility, if any, is available to the service provider in determining how the service should be composed. The author argues that the service provider should be given more flexibility and 'authority' in service composition. In the SBS approach, the service provider is given this flexibility by means of a flexible composition strategy of individual service provider, i.e. the *Know-How* capability (Fig. 1).

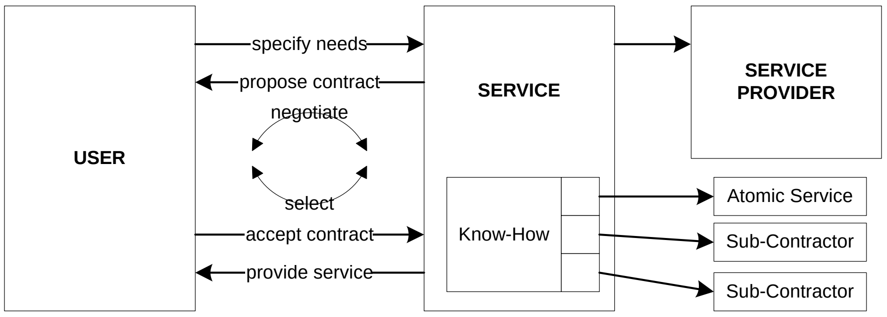

Fig. 1. Know-How Capability in Service-Based Software

This capability would enable service providers to choose the most appropriate service composition method (compose then offer services vs. offer then compose at execution). Therefore, the architecture for the software from individual services can be chosen prior to composition. To attain this capability, the software architecture style of services should be identified and determined, for example pipes and filters, prior to service composition.

In this paper the author proposes the use of the Alfa framework [1] to model the software architecture styles of services and to support the discovery of suitable services. Section 2 will present some background information on the SBS approach. Section 3 will present service architecture modeling using the Alfa framework. Section 4 will outline the expected results and contribution of this research. Finally, conclusion and future work will be presented in Section 5.

2 Background

This section will discuss some background information on Service Oriented Architecture, Service-Based Software, Software Architecture and the Alfa framework.

2.1 Software Oriented Architecture

Service Oriented Architecture (SOA) provides means to find, integrate, and use services that are available inside and outside the organization through the network/ Internet. SOA underlines three important issues for implementing Web services: interoperability, development and delivery. SOA identifies interoperability issues across open and proprietary platforms. This would allow organizations to benefit from a flexible IT architecture built on established standards without limiting the flexibility of business units to implement technologies that provide the capabilities they need. SOA also concerns the programming and development approach of services. Web services represent a programming approach that takes advantage of available objects and content that would enable faster development cycles and lower costs due to component reuse. In addition to that, SOA also deals with the openness and flexibility of service delivery where organizations can provide access to any type of information such as inventory transactions and financial data in a reusable and open fashion.

However, SOA limits the flexibility of the service provider to choose the method for service composition. Services are composed from a predefined and rigid architecture where sometimes the user's requirements could not be fulfilled completely.

2.2 Service-Based Software

The Service-Based Software (SBS) is based on the Software as a Service (SaaS) project, carried out by the Pennine group involving academicians from three UK universities. SBS is a new paradigm proposed by the group for software engineering, a service-based approach to structuring, developing and deploying software. SBS comprises of a number of small software components that meet user requirements. When a component needs to be updated or replaced; it is disengaged; suitable replacement is searched and acquired from the marketplace; integrated; and executed. SBS are acquired and paid for on demand, as and when needed. Research efforts in SaaS have been working on the principles and practicality of SBS. An architecture for SBS has been proposed and the feasibility of the concept and primitives of the SBS architecture has been demonstrated [2]. The basics of SBS have also been demonstrated using prototypes to perform service negotiation, composition, discovery, mediation and binding [3]. Studies in SBS also include the supplier-customer relationships [4], the negotiation process [5]. A 'Negotiation Description Language' has also been proposed for SBS [6].

SBS is a better approach for Web services compared to SOA as it gives the flexibility of identifying and determining the method for services composition.

2.3 Software Architecture

Software Architecture is generally referred to as the 'blueprint' of a software system that describes its coarse-grain components. Software architecture is often defined in terms of patterns of structural organization or *Software Architecture Style* (SAS) [7]. Each style has a vocabulary of components and a set of constraints on how they can be combined. SAS provides a collection of proven solutions to recurring design problems and demonstrates a method for integrating individual style into heterogeneous structures. SAS have been used informally and explicitly, and more than one style can be used. Examples of SAS include Pipes and Filters and Implicit Invocation. In addition to the styles in [7], there are also other styles found in literature for example [8],[9],[10],[11] and [12].

Clearly, there is wealth of software architecture styles that can be used in the service composition. However, these components need to be fine-grained and formalized before it can be used in the composition process.

2.4 Alfa

Alfa is an assembly language for software architecture and supports the modeling of data, processing, and connecting elements of an architecture [13]. However, Alfa goes beyond architectural description as it is based on a small set of architectural primitives with precise dynamic semantics. Alfa also supports comprehensive style conformance analysis and architectural composition using domain-independent primitives. Alfa's characterization of styles for composing architecture is adopted from the characterization of distributed styles using five dimensions: Data, Structure, Interaction, Behavior and Topology [14]. There are seventeen styles that have been characterized in Alfa including Client-Server and Layered systems.

Alfa's primitives have been shown to be expressive enough to model any interactions involving a regular expression of input/output events on point-to-point channels [13]. As such, Alfa can be used to represent the software architecture style for the service composition process.

3 Service Architecture Modeling Using Alfa

This section will present the idea of service architecture modeling using Alfa. The architecture of Web services can be modeled using one or more software architecture styles. For example, based on a set of user's requirements, the service provider decides that the best way to fulfill the requirements is by using *Event-Based Integration* (EBI) software architecture style.

Generally, EBI refers to a system where the components can announce or broadcast one or more events. Other components in the system can register an interest in any event by associating a procedure with it. When the event is announced, the system invokes all of the procedures that have registered for the event. As such, the system would have at least 3 components namely the *Publisher*, *Subscriber* and *Distributor* that announces, registers for, and manages the registration of events, respectively (please refer to Fig. 2).

Using Alfa's characterization of styles and primitives, this architecture is formalized and fine-grained to computational elements in the system. These would lead to the identification and determination of the data type; structure of the system and its components; topology of the system; interactions and behavior between the components. Using topology as an example, we could determine that the *Publisher's* publish interface is connected to the *Distributor's* accept interface; the *Subscriber's* subscription interface is connected to the *Distributor's* subscription interface; and the *Distributor's* distribute interface is connected to the *Subscriber's* consume interface.

Alfa's primitives provide information on the computational elements in the components and system. For example, RELAY is a connector that contains multiple INPUT and OUTPUT portals, and enables multi-point interaction. Each portal is housed in a separate INTERFACE. Data items from each INPUT portal are forwarded to every OUTPUT portal of the RELAY. DATUMs allowed at all the portals of a RELAY are identical (words in capital are Alfa's primitives).

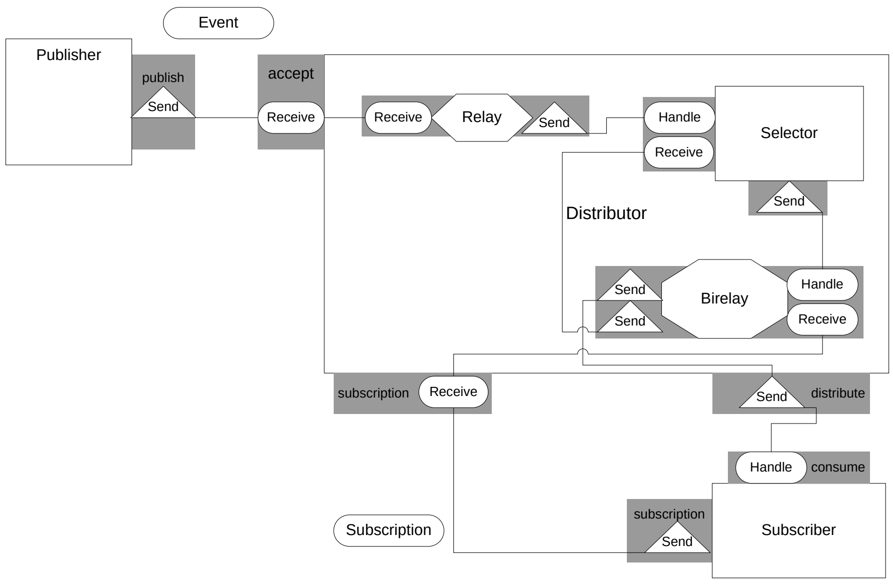

Fig. 2. Event-Based Integration Architectural Style Composed using Alfa

This information is also used for service discovery to fulfill specific requirements regarding the architecture of the service. Examples would be specific type of DATUMs, behavior of a specific PARTICLE, means of interaction either DUCT, RELAY or BIRELAY.

This section has presented an approach of using Alfa to model software architecture for the service composition process. For this approach to be implemented, software architecture and Alfa's primitives need to be included in the search criteria of Web services and the composition process.

4 Expected Results and Contribution

This research is still in an early stage where it proposes a more flexible approach to the composition of Web services by including the aspect of software architecture. In order accomplish this, at least two issues need to be addressed, a vocabulary of terms that will be used and the description of available services, both taking into account the software architecture aspects in the service composition process. Ontology of service architecture should be developed to ensure uniformity of terms used. Not only does this cover software architecture styles, but also the components in a specific style. Method for describing available services needs to be identified and developed in order for the services to be advertised and discovered. It is anticipated that the *Web Ontology Language* (OWL) and *Web Services Description Language* (WSDL) could be used for these purposes with the probability of extending WSDL to accommodate

Alfa’s primitives. A prototype would be developed at a later stage to demonstrate the feasibility of the whole SBS approach for Web services composition.

5 Conclusion and Further Work

This paper has outlined the SBS approach for Web services composition. The concept and use of software architecture and Alfa in for modeling and composing Web services were presented. Work in the near future will be investigating the use of OWL and WSDL in this research as discussed in the previous section.

References

1. Nenad Medvidovic, Nikunj R. Mehta, and Marija Mikic-Rakic. (2002). A family of software architecture implementation frameworks. In Proceedings of the 3rd IFIP Working International Conference on Software Architectures , Montreal, Canada.
2. Bennett, K. H., M. Munro, et al. (2001). An Architectural Model for Service-Based Software with Ultra Rapid Evolution. Proceedings of the 2001 International Conference on Software Maintenance (ICSM 2001), Florence, Italy, IEEE Computer Society.
3. Bennett, K. H., M. Munro, et al. (2002). Prototype Implementations of an Architectural Model for Service-Based Flexible Software. Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS-35 2002), Big Island, HI, IEEE Computer Society.
4. Brereton, P. (2004). "The Software Customer/Supplier Relationship." Communications of the ACM 47(2): 77-81.
5. Elfatatry, A. (2002). Service-Oriented Software: A Negotiation Perspective. PhD Thesis, UMIST.
6. Elfatatry, A. & P. J. Layzell (2004). "Negotiating in Service Oriented Environments." Communications of the ACM August 47(8).
7. Shaw, M. & D. Garlan (1996). Software Architecture: Perspectives on an Emerging Discipline. New Jersey, Prentice Hall.
8. Andrews, G. R. (1991). "Paradigms for Process Interaction in Distributed Programs." ACM Computing Surveys 23(1): 49-90.
9. Berson, A. (1992). Client/ Server Architecture, Mc-Graw Hill.
10. Mettala, E. and M. H. Graham (1992). The Domain-Specific Software Architecture Program. Technical Report CMU/SEI-92-SR-9, Carnegie Mellon Software Engineering Institute.
11. Tracz, W. (1994). Collected Overview Reports from the DSSA Project. Owego, Loral Federal Systems.
12. Harel, D. (1987). "Statecharts: A Visual Formalism for Complex Systems." Science of Computer Programming 8: 231-274.
13. Nikunj R. Mehta, Nenad Medvidovic. (2003). Composing Architectural Styles from Architectural Primitives. ESEC / SIGSOFT FSE. 347-350.
14. Fielding, R.T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis. University Of California, Irvine.