A Survey on QoS Based Service Composition for Service Oriented Architecture

P. Thangaraj, P. Balasubramanie

Abstract — Due to the tremendous growth in web services, Service Composition is a challenging problem due to the availability of composite services. Service selection is also a major factor that is to be considered during the retrieval of the service. These are influenced by functional and non functional parameters. The functional parameters include the integrity, behavior, fault tolerant service, etc. The non functional parameters include the efficiency, execution cost, time; availability; reputation and reliability are termed as QoS attributes. The various algorithms for composition include the heuristics and non heuristics. The problem formulated has to produce an optimized service composition and to select the efficient service that integrates and satisfies the needs of the end user.

Index Terms — web Service; Quality of Service; functional parameters; non functionl parameters; algorithm; Service Composition

I. INTRODUCTION

The Web services are distributed services that process XML-encoded SOAP messages, sent over HTTP, and described using Web Services Description Language (WSDL). Web services are used in a range of application integration scenarios: from simple, ad hoc, behind-the-firewall, data sharing to very large-scale Internet retailing and currency trading. And increasingly Web services are being applied in mobile, device, and grid scenarios. Web services provide interoperability between software components that can communicate between different companies and can reside on different infrastructures. This solves one of the most critical problems that customers, software developers, and partners face. HTTP and SOAP provide communication and message interoperability. WSDL provides the description of the service to support interoperability between development tools; it complements communication interoperability with the ability to exchange interface definitions. The main purpose of this paper is to briefly define the value these specifications provide to the end user. We also describe how these specifications complement each other to compose robust environments for distributed applications.

II. WEB SERVICES: A SERVICE-ORIENTED ARCHITECTURE

Web services to build an SOA, the solutions consist of collections of autonomous services, identified by URLs, with interfaces documented using WSDL, and processing well-defined XML messages. SOA is a natural complement to the object-oriented (OO), procedural and data centric approaches to solution implementation. Indeed, when creating an SOA system, the individual services are typically constructed using one or more of these technologies. A Service-Oriented Architecture differs from OO and procedural systems in one key aspect: *binding*. Services interact based on *what* functions they provide and *how* they deliver them. OO and procedural systems link elements together based on type or name. The following sections provide more detail.

2.1 Services are described by schema

The Web service model does not operate on the notion of shared types that require common implementation. Rather, services interact based solely on contracts (WSDL/BPEL4WS for message processing behavior) and schemas (WSDL/XSD for message structure). This enables the service to describe the structure of messages it can send and/or receive and sequencing constraints on these messages. The separation between structure and behavior and the explicit, machine verifiable description of these characteristics simplifies integration in heterogeneous environments.

2.2 Service compatibility is more than type compatibility

Service-orientation provides a richer model for determining compatibility. Structural compatibility is based on contract (WSDL and optionally BPEL4WS) and schema (XSD) and can be validated. Moreover, the advent of WS-Policy provides for additional automated analysis of the service assurance compatibility between services. This is done based on explicit assertions of capabilities and requirements in the form of WS-Policy statements. Using WS-Policy, services describe their service assurance capabilities and requirements in the form of a machine-readable policy expression containing combinations of assertions. This allows service to select each other based on "how" or "with what quality" they deliver their contracts,

2.4 Service-orientation enables flexible binding of services

One of the core concepts of *service-oriented architecture* (SOA) is flexible binding of services. An SOA supports more dynamic discovery of service instances that provides the interface, semantics and service assurances that the requestor expects.

Revised Manuscript Received on March 2015.

Prof. P. Thangaraj, CT- PG, Kongu Engineering College, Perundurai, Erode, Tamil Nadu, India.

Dr. P. Balasubramanie, Professor, Kongu Engineering College, Perundurai, Erode, Tamil Nadu, India.

In procedural or object-oriented systems, a caller typically finds a server based on the types it exports or a shared name space. In an SOA system, callers can search registries such as UDDI for a service.

1. That is message compatible with the caller's requirements. Compatibility can occur through WSDL or matching messages from well-known XML Schemas.
2. That documents support for service assurances that the caller requires. For example, the caller may desire certain approaches to security or transactions.

III. WEB SERVICE SPECIFICATIONS AND FUNCTIONS

3.1 A Composable Approach to Web Services

This section briefly describes the Web service specifications that are available. We explain their value to solution providers, their role in a broader architecture and how they complement each other. The following figure provides a high-level grouping of the Web service specifications published by IBM, Microsoft and others. Note that this figure is not meant to imply a strict layering between the groups; instead it is intended to provide an intuition about the relationships between functional areas. For example, message security does not require Description and similarly Description is a useful development time concept for Messaging.

3.2 The Basics—Transports and Messaging

The Interoperability of Web services faces the problem of interacting with other services. We address this by providing a common set of transports and messaging technology. Moreover, to ensure these technologies are effective in practice, IBM, Microsoft, and others created the Web Services Interoperability Organization (WS-I). Recently the WS-I released a basic profile that formally documents interoperable Web service transport and messaging mechanisms. This set of specifications defines the core communication mechanisms for moving raw data between Web services. HTTP, HTTP/S, and Simple Mail Transport Protocol (SMTP) are examples in this group.

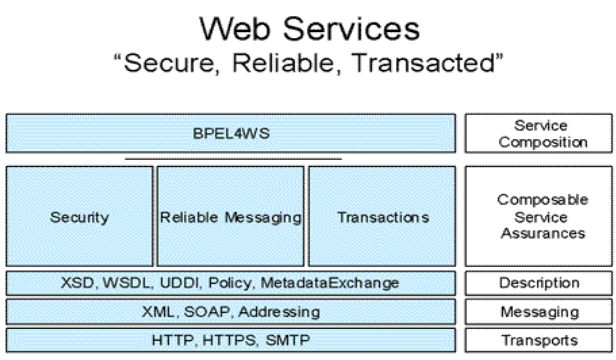

Figure 2. The interoperable Web services protocol architecture

The messaging specification group defines how to format messages properly. XML and XML Schema definitions provide the mechanism for abstractly agreeing on message (data) structures. SOAP defines the standard encoding for representing XML messages in the byte information that services exchange over transports.WS-Addressing provides an interoperable, transport independent approach to identifying message senders and receivers. WS-Addressing also provides a finer grain approach to identifying specific elements within a service that send or should receive a message. This approach builds on the basic browser-server model of HTTP. WS-Addressing decouples address information from any specific transport model.

3.3 Description

The transport and message specifications allow Web services to communicate using messages. The description group of specifications enables a Web service to express its interface and capabilities. In addition to message interoperability; these specifications also enable *development tool interoperability*. The description specifications provide a standard model that allows different tools from different vendors to collaboratively support developers. In the same way that Web services isolate partners from implementation and infrastructure choices, the description specifications isolate partners from development tool choices. The Web Services Description Language (WSDL) and the XML Schema (XSD) are the base specifications in this group. XML Schema allows developers and service providers to define XML types for data structures. WSDL provides support for a range of message interaction patterns. It supports one-way input messages that have no response, request/response, and one way sends with or without a response. The last two patterns enable a service to specify other services that it needs. The Proposed WSDL enhancements provide support for documenting protocols and message formats a service supports, and the service's address.

The WSDL and XSD define the service's interface syntax but they do not express information about how the service delivers its interface or what the service expects of the caller. WS-Policy enables a service to specify what it expects of callers and how it implements its interface. WS-Policy is critical to achieve interoperability at the higher-level functional operation of the service. Security, transactions, reliable messaging and other specifications require concrete WS-Policy schema. These allow services to describe the functional assurance that they expect from and provide to callers.

The XML, XSD, WSDL and WS-Policy support describing the interface and service assurances for a service. There are two options, the service may go directly to the service to obtain information using *WS-MetadataExchange* or it may choose to use a *UDDI* service that aggregates this information for multiple target services. The WS-MetadataExchange specification enables a service to provide metadata to others through a Web services interface.

Given only a reference to a Web service, a potential user can access a set of WSDL/SOAP operations to retrieve the metadata that describes the service. Clients can use WS-MetadataExchange at design time, when building their clients, or at runtime.

UDDI is useful to collect metadata about a collection of services and to make the information available in a form that is searchable. Such metadata aggregation services are a useful repository in which organizations can publish the services they provide, describe the interfaces to their services, and enable domain-specific taxonomies of services The developers may use these services in the definition of their BPEL4WS workflows, for example. Solutions can also query UDDI at runtime. In this scenario, the caller "knows" the interface it requires and searches for a service that meets its functional requirements or is provided by a well-known partner. Web services have generated so much enthusiasm in part because of their ability to bridge disparate systems. Developers have produced many fully functional solutions using the base capabilities of transport, messaging and description. However, to be accepted by developers creating more powerful integration solutions, Web services must provide functionality to ensure the same level of *service assurances* provided by more traditional middleware solutions. It is not enough to simply exchange messages. Applications and services reside in middleware and systems that provide valuable higher-level functions such as security, reliability, and transacted operations. Web services must provide a mechanism for interoperability between these functions.

3.4 Security, Messaging and Transaction

This family of specifications is critical to cross-organization Web services. These specifications support authentication and message integrity, confidentiality, trust, and privacy. They also support federation of security between different organizations. In an Internet world, almost all communication channels are unreliable. Messages disappear. Connections break. Without a reliable messaging standard, Web service application developers must build these functions into their applications. The basic approaches and techniques are well understood, for example many operating and middleware systems ensure messages have unique identifiers, provide sequence numbers, and use retransmission when messages are lost. If application Web service developers implement these models in their applications. They may make different assumptions or design choices, resulting in little if any reliable messaging. A complex business scenario may require multiple parties to exchange multiple sets of messages. An example is a set of financial institutions setting up a financial offering that involves insurance policies, annuities, checking accounts and brokerage accounts. The multiple messages exchanged between participants constitute a logical "task" or "objective" through WS-Coordination, WS-AtomicTransaction, and WS-BusinessActivity support these requirements.

3.8 Service Composition

The uppermost element in the Web service layering is *service composition*. Service composition allows developers to "compose" services that exchange SOAP messages and define their interface in WSDL and WS-Policy into an aggregate solution. The aggregate is a composed Web service.

The Business Process Execution Language for Web Services (BPEL4WS) specification supports service composition. It enables developers to define the structure and behavior of a set of Web services that jointly implement a shared business solution. Each element of the set of services defines its interface using WSDL and WS-Policy. The composed solution is itself a Web service, which supports HTTP/SOAP messages and defines its interface using WSDL and WS-Policy.

The Composition has three aspects: *structure*, *information* and *behavior*. BPEL4WS introduces three constructs supporting each composition aspect. A *partnerLink* defines a named association between the composite service and a Web service that participates in the overall solution. The composite service and participating service define their interfaces to each other using WSDL and WS-Policy. The partnerLink concept and the WSDL/WS-Policy interfaces between the composition and partners define the *structure* of the service composition. They define the types of services that collaborate to form the composition, and which messages they exchange with which levels of assurance (security, transactions, etc.)

To summarize, BPEL4WS makes two additions to the previously defined Web service specifications.

1. BPEL4WS extends WSDL and WS-Policy support for describing services. BPEL4WS support combining Web services into aggregate services, and documenting the associations between services, such as the information flow and behavior. This provides support for interoperability between higher layer tools supporting collaborative design of Web services.
2. BPEL4WS is an *execution language*. BPEL4WS allows developers to fully specify the behavior of a composite Web service. IBM, Microsoft, and other partners will provide environments that execute BPEL4WS documents and support design and execution time binding to partners.

The remarkable advances of practical Web services technologies that provide interoperability between distributed software components will enable different companies with different infrastructures to collaborate and communicate with one another. Web services are Web based enterprise applications that process SOAP messages described using WSDL and sent over HTTP, which use open XML-based standards and transport protocols to exchange data with calling clients. Service-oriented architecture (SOA) defines the use of loosely-coupled software services to

support the requirements of business processes and software users. Using such architecture, developers take services as fundamental elements in their application development processes. Web services technologies enable a rapid software development process by composing applications from replaceable and Internet accessible software components in a loosely-coupled, service-oriented manner. Thus, an important aspect is to ensure that the composite web service does not violate any non-functional properties i.e. QoS parameters. Parameters such as response time, availability, robustness, reliability and many others form user's experience and feedback indicating WSC efficiency. However, despite much research effort, state-of-the-art methods of Web service selection with QoS parameters taken into consideration cannot solve problem of WSS in complex, focusing only on narrow tasks. Such tasks as improving speed of composition or focusing on user preferences are important aspects, but solving one task and neglecting others is a problem which needs to be solved. In this paper an approach that overcomes above mentioned problems, as well as WSC System that performs QoS-aware web service selection are presented.

IV. CONCLUSION

Web Service Composition consists of two major blocks: Web Service Discovery, finding web services satisfying functional parameters and Web Service Selection, choosing the best possible combination of web services regarding functional parameters. Despite much research effort still many problems exist. WSD problem is lack of the syntactical description support in fuzzy logic algorithms. WSS problem is narrow task focusing and thus neglecting of other important aspects. WSC System covers such aspects as full stack of QoS parameters support, subjective QoS i.e. user preferences, and monitoring stage support. Another important issue is synchronous utilization of WSD and WSS. It means that presented approaches are fully compatible. After the integration, comprehensive system testing will be applied and quantitative results provided. Also tutorial for WSC System is to be written.

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in American English is without an "e" after the "g." Use the singular heading even if you have many acknowledgments. Avoid expressions such as "One of us (S.B.A.) would like to thank ... ." Instead, write "F. A. Author thanks ...." **Sponsor and financial support acknowledgments are placed in the unnumbered footnote on the first page**.

REFERENCES

1. Mardukhi, F., NematBakhsh, N., Zamanifar, K., Barati A.: QoS decomposition for service composition using genetic algorithm. In: Applied Soft Computing, Volume 13, Issue 7, pp. 3409--3421.
2. Wei Z., Junhao W., Min G., Junwei L.: A QoS preference-based algorithm for service composition in service-oriented network. In: Optik - International Journal for Light and Electron Optics, Vol. 124, Issue 20, pp. 4439--4444.
3. Curbera F., Duftler M., Khalaf R., Nagy W., Mukhi N., Weerawarana S.. "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI," Internet Computing, IEEE, Vol. 6, pp. 86-93, 2002
4. Küster, U., H. Lausen, and B. König-Ries, Evaluation of Semantic Service Discovery—A Survey and Directions for Future Research. Emerging Web Services Technology, 2008. 2: p. 41-58.
5. Keyvan M., Suhaimi I., Mojtaba K., Kanmani M., Sayed G. H. T.i, A comparative evaluation of semantic web service discovery approaches, Proceedings of the iiWAS2010, November 08-10, 2010, Paris, France.
6. Klusch, M., Semantic Web Service Coordination, in CASCOM: Intelligent Service Coordination in the Semantic Web. 2008. p.59-104.
7. Klusch, M., B. Fries, and K. Sycara, OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services. Web Semantics: Science, Services and Agents on the World Wide Web, 2009. 7(2): p. 121-133
8. Kourtesis, D. and I. Paraskakis, Combining SAWSDL, OWLDL and UDDI for Semantically Enhanced Web Service Discovery, in The Semantic Web: Research and Applications. 2008. p. 614-628
9. Klusch, M., P. Kapahnke, and I. Zinnikus, SAWSDL-MX2: A Machine-learning Approach for Integrating Semantic Web Service Matchmaking Variants, in IEEE International Conference on Web Services. 2009.
10. Duy Ngan Le ; Goh, A.E.S. FuzMOD: A Fuzzy Multi-ontology Web Service Discovery System. APSCC, The 2nd IEEE. 2007, pp. 197-203
11. Srinivasan, N., M. Paolucci, and K. Sycara. Semantic Web Service Discovery in the OWL-S IDE. in System Sciences. Proceedings of HICSS '06, 2006
12. Somasundaram, T.S., et al. Semantic Description and Discovery of Grid Services using WSDL-S and QoS based matchmaking Algorithm. in ADCOM 2006
13. Li, H., X. Du, and X. Tian, A WSMO-Based Semantic Web Services Discovery Framework in Heterogeneous Ontologies iiWAS2010 Proceedings Web Services Environment. Lecture Notes In Computer Science, 2007. 4798: p. 617.
14. Pukhkaiev, D., Kot, T., Globa, L., Schill, A.: A novel SLA-aware approach for web service composition. In: IEEEEUROCON, pp. 327--334 (2013)
15. Berbner, R., Spahn, M., Repp, N., eckmann, O., Steinmetz, R.:Heuristics for QoS-aware Web Service Composition. In: Proceedings of the IEEE International Conference on Web Services 2006, pp. 72--82. IEEE Computer Society Washington, DC(2006)
16. Aiello, M., el Khoury, E., Lazovik, A., Ratelband, P.: Optimal QoS-Aware Web Service Composition. In: IEEE Conference on Commerce and Enterprise Computing, pp. 491—494 (2009)
17. Kot T., Reverchuk A., Globa L., Schill A.: A Novel approach to increase efficiency of OSS/BSS workflow planning and design. In: Proceedings of BIS'12,vol.117,pp.142-15.