A SUCCINCT ANALYSIS OF WEB SERVICE COMPOSITION

Wassam Zahreddine, Qusay H. Mahmoud
Department of Computing and Information Science
University of Gueph, Guelph, ON, N1G 2W, Canada

Keywords: Web service composition, BPEL4WS, OWL-S, WSCI, WSCL, software agents, P2P.

Abstract: Numerous standards are being proposed by industry and academia to find ways to best compose web services together. Such standards have produced semi-automatic compositions that can only be applied in a limited number of scenarios. Indeed, the future is moving towards a semantic web and fully automatic compositions will only occur when semantics are involved with web services. This paper presents brief notes on the state of the art in the field of service composition. The paper classifies service composition into two streams: semi-automatic and fully automatic, then compares and contrasts the available composition techniques.

1 INTRODUCTION

Web services are becoming an attractive solution for businesses and consumers alike because of their simplicity and reusability. Presently, web services are designed to be modular and loosely coupled to perform a specific set of operations such as retrieving a stock quote. However, what if a client requires a service that no one web service can satisfy? The modularity of web services has created a composition problem that is still in need of an optimal solution. Web service composition involves an amalgamation of two or more web services to fulfil a request that no one web service is able to provide. The endless possibilities of a composite web service will bring forth a new wave of online applications.

Web services are designed to perform a specific set of operations and ideally perform them well. Presently, the user would pick and choose the organization that offers a web service that performs optimally, or for the best price, or any other required criteria. If the user requires more than what any one particular web service has to offer, they can manually invoke other web services and organize the results on their own. However as the need for various web services grow so does the complexity of choosing the appropriate service and managing the results. The time involved in searching, selecting, and invoking individual web services could be immense depending on the number of web services needed. Having the ability for an application to perform these tasks would be ideal. Therefore, there needs to be some tools to manage and ensure that services perform correctly together.

2 SEMI-AUTOMATIC METHODS

There have been various attempts by industry to build composite web services. This section will focus on the many important technologies developed by industrial efforts, namely: BPML, WSCI, WSCL, BPEL4WS, XLANG, and WSFL.

2.1 BPML & WSCI

The Business Process Management Language (BPML) provides an abstract model and grammar for describing business processes and was developed by Business Process Management Initiative (BPMI.org). Initially it was designed for the BPMS system to support business processes. However, the first draft used the Web Services Choreography Interface (WSCI), this is an extension of WSDL to describe the behaviour of a web service and the flow of messages. BPML handles the orchestration of web services while WSCI handles the choreography between web services. Hence, WSCI only describes the visible behaviour and not the definition of executable business processes as BPEL4WS does (Peltz, 2003) but BPML makes up for that with its own orchestration techniques. The BPML model is a composition of activities that represent a business process. Each activity performs a specific function (a unit of work) and the process directs how and

Zahreddine W. and H. Mahmoud Q. (2006).
A SUCCINCT ANALYSIS OF WEB SERVICE COMPOSITION.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - SAIC*, pages 193-196
DOI: 10.5220/0002495301930196

when these activities are executed. This approach is very similar to UML activity diagrams.

Moreover, during a message exchange, only one partner's participation is described in a single WSCI interface and is therefore designed from the perspective of one side of the partnership (Peltz, 2003). Realistically, you need to map the actions of both sides during collaboration. WSFL had already accounted for this scenario with the concept of a global model. Later on in the development of WSCI global models were included. The formation of a global model using WSCI would be assuming the two parties know a lot about each other's process. A bad approach considering the global model is assumed immutable. This made BPML difficult to use for B2B (Business to Business) scenarios and was likely one of the main reasons why the WSCI protocol was removed from the BPML 1.0 specification (but it is still supported).

2.2 WSCL

Web Services Conversation Language (WSCL) is a service description language using a simple state-transition model for organizing the sequence of WSDL operations. WSCL can be used to describe service interactions and to specify a web service interface. Also this protocol orchestrates the message exchanges that occur at each stage of the conversation. WSCL portrays the conversation pattern that a web service will be engaged in by describing the order is which WSDL operations should be invoked. WSCL has the following basic concepts that describe conversations: DocumentTypes to reference XML Schemas, Interactions for one or two way message exchanges, and lastly Transitions to describe how to move from one interaction to another. It is expected that WSCL will be extended to describe more complex scenarios such as multi-party conversations and composition, service attributes, and transactions. Unlike other modelling languages it does not support contexts, exception handling, time-outs, and contexts.

2.3 BPEL4WS, XLANG & WSFL

One modelling language with major industrial backing is BPEL4WS (Business Process Execution Language for Web Services) which evolved from the amalgamation of Microsoft's XLANG and IBM's WSFL (Web Services Flow Language). This new language combines the best features of WSFL and XLANG. The merge involves taking the graph oriented, transitioned-based process representation of WSFL and the block structured processes of XLANG. Together they build a new orchestration language defining interactions between web services. WSFL supports two model types namely flow and global models. The flow model describes business processes that use predetermined set of web services. The global model describes how web services will interact with each other. And XLANG orchestrates how individual web services become a business process and composite web service.

Furthermore, BPEL4WS was also released with two other specifications, WS-Coordination and WS-Transaction. WS-Coordination defines a framework that allows different coordination protocols to manage operations between participants. WS-Transaction allows businesses to monitor coordinated activities in a business process. These will ensure that all the transactions complete successfully or fail as a group. In addition, the model has scopes and handlers to manage exceptions and allow alternative actions to be taken or reverse work in a previously completed scope (Curbera et al, 2003). This language scales well and offers primitive constructs of the language such as 'sequence' and 'while'. Even more interesting is when the composition is advertised as a web service, other compositions may include it their workflow. Furthermore, BPEL4WS also gives the composer flexibility in developing a workflow since two different styles of modelling are supported: the graph-oriented style of WSFL and the block structured algebraic style of XLANG (Curbera et al, 2003). The merging of these two languages also means that BPEL4WS supports all the patterns that WSFL and XLANG, however this flexibility comes at the price of a language with greater complexity. Especially when considering the overlapping constructs and the compromises that had to be made between Microsoft and IBM to create the language.

Moreover, BPEL4WS is still in its infancy and is the first step towards building a technology that may become the industry's first choice in web service composition. Furthermore, BPEL4WS is semi-automatic although it does support "runtime binding", where the parties involved in the composition is not known until they are needed. Realistically, it can dynamically assign a partner not dynamically discover new partners. Binding to service partners depends on the descriptions of what the services do and how they work, this is done by references to the <portTypes> in WSDL. Unfortunately, since WSDL lacks semantics it fails to describe how the web service works or what it does, and only provides the methods, parameters, and return values. In this manner, service behaviour is restricted by XML making it a difficult decision on whether to bind with a service partner or not.

Many models have been presented here however they all fall in relatively the same scope. These languages focus on a syntactical approach and can only produce semi-automatic composition. What it

really comes down to is having a person in the loop taking care of the workflows and business processes. This may suffice for services that rarely change or knowing *a priori* the relationships between business partners. However the growing number of web services will create new options that may entice new business partnerships, hence complicating the building of workflows.

3 AUTOMATIC COMPOSITION

The automation of web services is difficult with standard WSDL descriptions because they do not provide any machine comprehendible metadata to assure that the service is being used correctly. More importantly a machine cannot understand whether a web service will perform the needed service or not. The liability is on the user to ensure the appropriate web service is selected.

3.1 Semantic Web

Presently, web sites are designed to describe their appearance. However, as automation becomes more important, web pages will have to describe their meanings as well.

Searches based on syntactical means fall victim to false positives when the word is a homograph or false negatives when the word is a synonym. Researchers are looking for solutions to solve the semantically deprived Internet. This has led to the creation of semantic languages that are meant for computers, and not for humans. RDF (Resource Description Framework) is one of the first languages for representing information about web resources. However, RDF lacked the expressiveness needed and the semantics themselves were underspecified. Then OWL (Web Ontology Language) was developed on top of RDF to support processing information on the web. Currently, web services are defined by RDF using RDFS (RDF Schema). Other languages such as DAML-ONT, OIL, and DAML+OIL have emerged as a way to improve the semantic richness of web resources. These languages are written in XML and describe UDDI and WSDL documents.

Moreover, it is important to ensure that the ontology being used is standardized so that all web services follow the same ontology. One unpleasant scenario would involve invoking a web service with non-standardized ontology that delivers unintended material or even spam. Therefore, not only does the web service need to follow a standardized ontology but it must also have the means to verify that the service is indeed following a standard. Security issues with ontology-based systems still need to be addressed. And what were to happen if a standard ontology changes? The affected web service would have to redefine its ontology and ensure it complies with the updated standard.

3.2 OWL-S/DAML-S

A major player in web service automation is the DAML (DARPA Agent Markup Language) organization. Their initiatives in web services have produced DAML-S, and now its successor OWL-S. This language is built upon OWL and its predecessor ontology, DAML+OIL. These languages are based on the DARPA knowledge sharing format and works by restricting services to use a common syntax, ontology, and protocols. OWL-S has three properties to describe a service. The ServiceProfile and ServiceModel are the abstract representations of a service and ServiceGrounding deals with the concrete level of specification. The ServiceProfile describes the capabilities and parameters of the service. The service model describes what happens when the service is carried out. More formally, it describes how the service works by specifying the workflow and possible execution paths. The service grounding explains how the service can be accessed and used. Grounding will specify the protocol to use, message types and other details specific to the service. Orchestration by OWL-S is performed similarly to other choreography languages such as BPEL4WS. It has control constructs such as ‘sequence’, and ‘if-then-else’ which form the composition plan.

3.3 Software Agents

A software agent is a program that performs a specific task on behalf of the user or another agent. Agents can autonomously solve problems, and are goal-oriented. The agent is more commonly created from an agent platform, where it then moves from one platform to another invoking the methods accessible to the agent. Unlike remote method invocation, the state of the agent can be stored and can continue where it left off on the new platform. A mobile agent is extremely useful in the field of wireless computing because they improve latency and the use of bandwidth since the invocations are handled locally on the hosts' platform. Agents can also communicate with other agents and act as autonomous communicative middleware. For instance, an agent that acts as a yellow page to find other agents.

The use of agents with web services is interesting because of the benefits agents bring to web services (Zahreddine et al, 2005). For instance, web services are stateless and know only of

themselves; conversely agents can be personalized, adaptable, and context-aware. Also, agents can act as a middleware for composition, manage results and exception handling. However, it is important to note that not every agent-based approach supports fully automatic composition. In order to have automatic composition, an intelligent discovery service is required to find the right web services and understand how to piece them together.

Work in (McIlraith et al, 2001) is one of the initial discoveries that involve using agents and semantic web services together. Their work has taken the ordinary web services and wrapped them in a DAML-S ontology and shown how agents can successfully perform automatic web service composition. A Plan Domain Description Language (PDDL) is used as the artificial intelligence planner for web service choreography. Using the control constructs provided by DAML-S the planner decides how and when to invoke web services. Agents written in ConGolog perform the actual invocations.

Furthermore, using agents in composition does have its disadvantages. Building and testing an agent society is difficult. There are also security risks when using agents. Firstly, protecting the host from malicious agents is imperative. Many techniques have been documented to solve this problem such as: authentication and authorization (Berkovitz et al, 1998) or sandbox approaches that limit privileges. Secondly, protecting agents from malicious hosts; this facet of security has not been solved by software means. Some believe the answer is in a mutually trusted third party (Algesheimer et al, 2000). These security fears are important set backs in software design and are one of the reasons why agents do not have a wide spread employment over the net.

3.4 P2P

Another approach involves the field of P2P (peer to peer) computing such as in (Benatallah et al, 2003; Pitoura et al, 2003; Abiteboul et al, 2002). In particular, the P2P orchestration model Self-Serv (Benatallah et al, 2003) is an interesting approach towards a middleware infrastructure for web service composition. The Self-Serv model proposes that composition of web services through a decentralized dynamic environment. Self-Serv brings together elementary and composite services. An elementary service is an individual web service that does not rely on other web services. A composite service is referred to as a component and aggregates more than one web service together; business logic is expressed as a state chart.

4 CONCLUSION

Semi-automatic compositions can only be applied in a limited number of scenarios. The future is moving towards a semantic web and fully automatic compositions will only occur when semantics are involved with web services. Automatic web service composition does have its costs, such as in service discovery. Searching a UDDI takes time, especially when dealing with multiple registries. Although efforts in the P2P field have helped, and knowing the parties you are dealing with at design time rather then runtime is quicker. However, this form of composition is only appropriate if you know what web services you need, know how to use them, and that they rarely change. Realistically, there will always be circumstances when services need to be discovered at runtime especially when automatic composition is needed, and as the number of web services grows.

REFERENCES

Abiteboul, S., Benjelloun, O., Manolescu, I., Milo, T., Weber, R., 2002. Active XML: Peer-to-Peer Data and Web Services Integration. VLDB, pp. 1087-1090.

Algesheimer, J., Cachin, C., Camenisch, J., and Karjoth, G., 2000. Cryptographic Security for Mobile Code. Technical Report RZ 3302 (# 93348), IBM Research.

Berkovitz, S., Guttman, J.D., and Swarup, V., 1998. Authentication for Mobile Agents. Lecture Notes in Computer Science, Vol. 1419.

Benatallah, B., Sheng, Q.Z., Dumas, M., 2003. The Self-Serv Environment for Web Services Composition, IEEE Internet Computing, Vol 7 No 1, pp. 40-48.

Curbera, F.C., Khalaf, R.Y. and Leymann, F., 2003. Composing Web Services Using BPEL4WS, OMG Web Services Europe 2003: Web Services for Integrated Enterprise.

McIlraith, S., Son, T.C., and Zeng, H., 2001. Semantic Web Services, IEEE Intelligent Systems. Special Issue on the Semantic Web. 16(2):46-53.

Pitoura, E., Abiteboul, S., Pfoser, D., Samaras, G., Vazirgiannis M., 2003. DBGlobe: A Service-Oriented P2P System for Global Computing, Sigmod Record, SIGMOD Record 32(3): 77-82.

Peltz, C., 2003. "Web Services Orchestration and Choreography", IEEE Computer, Vol. 36, No. 10, pp 46-52.

Zahreddine, W., and Mahmoud, Q.H., 2005. Blending Web Services and Agents for Mobile Users. Proc. of the 7th ISADS Workshop on Cooperative Computing, Networking, and Assurance, Chengdu, China, pp. 585-590.