Towards an Efficient, Reliable and Collaborative Web: from Distributed Computing to Semantic Description, Composition and Matchmaking of Services

Adomas Svirskas, Michael Wilson, Brian Matthews, Alvaro Arenas, Damian Mac Randal, Julian Gallop, Juan Bicarregui, Simon Lambert

CCLRC Rutherford Appleton Laboratory

Abstract

Any protocols for loosely coupled, open web services must support both fast performance and scalability to very large numbers of services and service interactions. Service discovery, composition, and contract/SLA management require a representation language which is both sufficiently expressive and capable of graceful degradation outside its core application scope, whilst meeting the general performance and scalability requirements. Existing distributed technologies, and proposed Semantic Web technologies are reviewed with respect to these requirements. The paper

1. BACKGROUND: HOW WE HAVE GOTTEN HERE

The architectures of distributed systems and applications, along with the users' perception and expectations have changed radically during the last 10-15 years. A significant milestone was set when distributed object environments (primarily OMG's *Common Object Request Broker Architecture - CORBA* [1]) made it possible to develop distributed applications as if the remote objects were local. The most popular mass-market examples of component-oriented middleware were Microsoft's COM family of technologies [2], DCOM in particular, which was competing initially with CORBA and later with Sun's EJB[3], a part of J2EE [4]. The Microsoft .NET Platform as a programming model initially represented the next stage in the evolution of COM [11], however at the moment Microsoft markets .NET as a Web Services platform [12].

While the impact of these technologies to the software industry was tremendous, during a decade of experience both researchers and practitioners found out that distributed object computing has inherent problems, primarily because of fairly tight coupling required between the parties.

After the advent of the Internet and, especially after the Internet became the vehicle of preference for integrating corporate applications, each end every middleware technology has been re-assessed from the Internet-compatibility point of view:

- How efficient it is, given the fact that public networks and the Internet in particular differ from private in many aspects
- How well it maps to the standards de facto of the Internet – HTTP, SMTP etc.
- How scalable it is when the number of users/clients is not regulated by a single authority.
- How costly is discovery of potential communication partners
- How secure it is when used in the Internet environment and how expensive is to maintain consistent security for this solution

- How much additional requirements it puts on the general Internet-related security measures, i.e. how firewall policies are affected

The success if the Internet and the Web, which are based on uniform communication protocols, has shown that tightly coupled software systems are only good for niche markets and proprietary solutions, whereas loosely coupled software systems can be much more appropriate in practice due to its flexibility and openness, defined by discoverable public interfaces. Loose coupling makes it easier for systems (legacy or not), which do not share common assets, to interact between themselves.

2. SERVICE ORIENTED ARCHITECTURE AND WEB SERVICES

2.1 General Considerations

One way to conceive of such a loosely-coupled system is to consider it as composed of a collection of interacting *services*. Brown et al. [6] give a definition of service: "*A service is generally implemented as a course-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled (often asynchronous), message-based communication model*" and relate component-based development (CBD) to SOA by arguing that "*components as the best way to implement services, though one has to understand that an exemplary component-based application does not necessarily make an exemplary service-oriented application*". Brown et al. [6] outline the transition roadmap from the CBD to SOA encouraging reusing of CBD models, where possible and applying new SOA patterns, where necessary. This view is supported by Borges et al. [9], who claim that service modelling is about identifying the right services, organizing them in a *manageable hierarchy of composite services (smaller grained often supported larger grained), choreographing them together for supporting a business process*. We will return to this particular point of view in the subsequent sections. Borges et al. [9] see in SOA one key differentiation with more traditional OO/CBD in the sense that one no longer creates large object models, but rather designs the internals of larger-grained, business-aligned service boundaries, through finer-grained component -orientation. When *properly implemented*, services can be discovered and invoked dynamically using non-proprietary mechanisms, while each service can still be implemented in a black-box manner.

2.2 Architectural Requirements for Web Services

The considerations above touch only the basic aspects of interoperability, leaving aside the whole range of enterprise-grade "ilities"- security, manageability, discoverability, reliability, quality of service, crash recovery etc [5]. Therefore, there is a need for distributed computing solutions offering feature-rich environments (such as CORBA Services and Facilities), which would be as efficient as component-based frameworks (e.g. J2EE) and as interoperable as the basic Web protocols (e.g. HTTP). Web Services, being the technology which fulfils interoperability requirement must be enhanced appropriately to meet the mentioned needs.

In addition to this, there are specific requirements related to e-business and Virtual Organisations, these include choreography of services, on-demand discovery, flexible and efficient match-making, dynamic policy re-configuration and management, security, trust and privacy control.

For example, if we return to Borges' [9] claim about the need for manageable hierarchy of composite services (smaller grained often supported larger grained), choreographing them together for supporting a business process, we can clearly see that rich service description and explicit choreography of services are very important to achieve said goals. Business collaboration (potentially in a VO) occurs between peers and takes form of message exchange between them, according to a defined set of rules. These are global "reactive" rules that declaratively prescribe normal/abnormal progress, common agreement of the outcomes and are used by each participant to determine which message exchange will/can hap-pen next at any point of collaboration. A set of said rules collectively is referred to as choreography, which also provides a definition of the information formats being ex-changed by all participants. Through the use of a global model, choreography ensures that contractual behaviour across multiple services can be achieved without complex wiring or complex wiring tools [15]. Recursive composition model that lets you build choreographies incrementally by combining existing choreographies, which is necessary to address complex inter-organization business processes [14].

The key differentiator between orchestration and choreography is the presence or absence of a "conductor". Orchestration implies centralized control, a delegation (abandonment) of responsibility for achieving the overall goals to the conductor, who can command the services and handle errors/problems that arise to ensure the efficient achievement of the overall goals. This places a smaller load on the individual services (which is why organizations tend to be hierarchical), but is inherently inflexible and dependent on the conductor. Conversely, choreography relies on each individual service to respond appropriately when errors/problems occur, and this requires each service to understand the overall goals and the overall context (hence semantics). This places a greater load and responsibility on the services

Flexible, efficient and accurate service discovery and matchmaking are needed at run-time when collaborating parties, taking chosen roles in defined choreographies, attempt to fulfil their obligations and implement their interfaces. These aspects will be addressed in the sections below.

3. SEMANTIC WEB ROLE IN WEB SERVICES

While the Advanced Web Services Architecture (WSA) [8] addresses many important requirements and is definitely a big step towards better Web-based middleware, there are a number of aspects, which are not addressed and, perhaps, can't be addressed using the "classical" Web Services ingredients, such as SOAP, Web Service Description Language (WSDL), Universal Description, Discovery and Integration (UDDI). As an example, we can take a program, which must be able to automatically find, or discover, an appropriate Web service as a part of choreography realisation. Neither WSDL nor (UDDI) allows for software to determine what a Web service offers to the client. To make use of a Web Service, a software agent needs a computer-interpretable description of the service and the means for access, advanced matchmaking services, which are necessary for efficient discovery, require rich and flexible metadata that are not currently supported by UDDI. Furthermore, if we think of a closer integration of Web Services and the Web (e.g. HTML), there are some issues - while SOAP Web Services use the HTTP infrastructure, they are not able to hyperlink SOAP Web Service through HTML links or XSLT functions [16].

When it comes to rich and flexible metadata, description of properties, capabilities and interrelations between the entities, the Semantic Web [17] is quite obvious choice, as Berners-Lee et al. write "*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*"

A Semantic Web service describes its properties and capabilities so that software can automatically determine its purpose. An important goal for Semantic Web mark-up languages is to establish a framework for making and sharing these descriptions. Web sites should be able to employ a set of basic classes and properties for declaring and describing services, and the ontology structuring mechanisms of OWL provides the appropriate framework to do this [16]. Alesso [16] further argues that ontology for Web Services would make Web Services machine understandable and support automated Web Service composition and interoperability. Therefore, the motivations for creating OWL-S include ontology for Web Services that provides several automated functions: *service, service invocation, service composition, service monitoring*.

A Semantic Web service provides a descriptive list of what an agent needs to be able to do to execute and fulfil the service. This includes what the inputs and outputs of the service are. Software must be able to select and combine a number of Web services to complete a certain objective. The services have to interoperate with each other seamlessly so that the combined results are a valid solution. In this way, agent software can *create entirely new solutions*, which are part of on-demand business collaborations. Agent software needs to be able to verify and monitor the service properties while in operation. We need to be able to program agents to locate Web Services which satisfy a set of constraints [16].

The Web Services Modelling Ontology (WSMO) [19] and the Semantic Web Services (OWL-S) [20] are the most notable initiatives to describe Semantic Web Services in order to enable the automation of Web Service discovery, composition, interoperation and invocation. The WSMO framework has an important feature – it provides support for choreography and orchestration as part of interface definition of a WSMO service description. An interface describes how the functionality of the service can be achieved (i.e. how the capability of a service can be fulfilled) by providing a twofold view on the operational competence of the service: *Choreography decomposes a capability in terms of interaction with the service* (service user's view) *Orchestration decomposes a capability in terms of functionality required from other services* (other service providers' view) With this distinction we provide different decompositions of process/capabilities to the top (service requester) and to the bottom (other service providers). This distinction reflects the difference between communication and cooperation. The choreography defines how to communicate with the web service in order to consume its functionality. The orchestration defines how the overall functionality is achieved by the cooperation of more elementary service providers [19].

Revisiting our point about modelling business collaboration in VOs it is worth noticing that WSMO Choreography and WS-CDL share certain concepts, however WSMO support for choreography and orchestrations needs to be explored more thoroughly in order to maximize the impact of both Web Services semantic descriptions and the WS-* specifications, such as WS-CDL, proposed by software industry and maintained by W3C.

In taking such information integration and aggregation to a new level, the Semantic Web creates new trust and security challenges. Comprehensive Semantic Web Trust and Security Framework is needed cope with challenges by addressing three aspects: trust, identity, and signatures.

One of the goal of is to allow a Semantic Web application to reason about which combination of information sources they trust enough to use for a particular purpose that it is currently engaged in, where trust is trust for the task at hand, rather than a simple binary differentiation between trusted and untrusted parties. Vocabularies and ontologies for describing trust relevant metadata are needed for such reasoning. This metadata will differentiate between assertion and quoting as two important primitives.

Confidence in particular assertions can be greatly enhanced by confirming the identity of the parties involved. By moving identity information from digital signatures and public key certificates into the realm of discourse, the concepts from public key infrastructures can be integrated with a wider notion of trust on the Semantic Web that encompasses a wide range of other metadata. The integrated framework can be applied to specific use cases including the annotation of Web pages, Web service description and the large scale integration of scientific data.

4. CONCLUSIONS

On a concluding remark we must point out that the Semantic Web initiatives in Web Services area are quite significant and it is important to align SW-based developments with the WSA specifications. We think that is especially important to leverage and strengthen Semantic Web and Web Services synergy in the areas, where rich models are needed to represent service capability description, service matchmaking, dynamic service composition, quality of service, security and trust.

5. REFERENCES

[1] CORBA FAQ, http://www.omg.org/gettingstarted/corbafaq.htm
[2] COM: Component Object Model Technologies, http://www.microsoft.com/com/default.mspx
[3] Enterprise JavaBeans Technology, http://java.sun.com/products/ejb/
[4] Java 2 Platform, Enterprise Edition (J2EE), http://java.sun.com/j2ee/index.jsp
[5] Filman, R. E. (1998). "Achieving Ilities," Workshop on Compositional Software Architectures, Monterey, California, Jan. 1998. http://ic.arc.nasa.gov/people/filman/text/oif/wcsa-achieving-ilities.pdf
[6] Brown, A. et al. "Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications". A Rational Software Whitepaper from IBM. TP032, April 2003, http://www-128.ibm.com/developerworks/rational/library/content/03July/2000/2169/2169.pdf
[7] Ferguson, D.F., Storey, T., Lovering, B. and Shewchuk, J. (2003). Secure, Reliable, Transacted Web Services http://www-106.ibm.com/developerworks/webservices/library/ws-securtrans/
[8] Chiusano, J., M. and Booz, A., H. (2003). Web Services Security and More: The Global XML Web Services Architecture (GXA) http://www.developer.com/services/article.php/2171031
[9] Borges, B., Holley, K., Arsanjani, A. Delving into Service-Oriented Architecture http://www.developer.com/java/ent/article.php/3409221
[10] Web Services Architecture W3C Working Group Note 11 February 2004, http://www.w3.org/TR/ws-arch/
[11] Hoagland, J. .NET Platform as Component Infrastructure, http://www.components-online.com/NETPlatform/default.htm
[12] Basics of .NET, http://www.microsoft.com/Net/Basics.aspx
[13[Web Services Activity, http://www.w3.org/2002/ws/
[14] Afshar, M., Hilderbrad, H., Kavantzas, N. Shaffer, D. Surpur, A. (2004) Process-centric realization of SOA: BPEL moves into the limelight Web Services Journal, Nov,2004 http://www.findarticles.com/p/articles/mi_m0MLV/is_11_4/ai_n7071401/pg_1
[15] W3C's WS-CDL Targets Peer-to-Peer Web Services Collaboration, http://www.intldeveloper.co.uk/news/business+news/w3c+targets+web+services.asp
[16] Alesso, H. P. Developing the Next Generation Web Services - Semantic Web Services, http://www.webservicessummit.com/Excerpts/BuildingSemanticWS.htm
[17] Semantic Web, http://www.w3.org/2001/sw/
[18] Tim Berners-Lee, T., Hendler, J., Lassila, O. The Semantic Web, Scientific American, May 2001 http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2
[19] D2v02. Web Service Modeling Ontology - Standard (WSMO - Standard) WSMO Working Draft 06 March 2004, http://www.wsmo.org/2004/d2/v0.2/20040306/
[20] OWL-S 1.1 Release, http://www.daml.org/services/owl-s/1.1/
[21] D14v0.1. Ontology-based Choreography and Orchestration of WSMO Services http://www.wsmo.org/TR/d14/v0.1/