Towards a Semantic Event-Based Service-Oriented Architecture

Carlos Pedrinaci[1], Matthew Moran[2], and Barry Norton[1]

[1] Knowledge Media Institute, The Open University, Milton Keynes, UK.
{c.pedrinaci, b.j.norton}@open.ac.uk
[2] Digital Enterprise Research Inst. (DERI), National University of Ireland, Galway.
matthew.moran@deri.org

Abstract. [1] Service-Oriented Architecture (SOA) is commonly lauded as a silver bullet for Enterprise Application Integration, inter-organizational business processes implementation, and even as a general solution for the development of all complex Web-oriented applications. However, SOA without semantic descriptions of its data, processes and messaging models fails to achieve a truly flexible and dynamic infrastructure. In this paper we explain where semantics are necessary for SOA and present early work on a Semantic Execution Environment which couples the Service-Oriented and Event-Driven architecture styles with formal semantics.

1 Introduction

Service Oriented Architecture (SOA) is currently regarded as the next step for software architectures, catering for the assembly of software systems into more complex solutions with the promise of reasonable costs. Although WSDL, XML and BPEL make SOA possible, problems quickly arise when services using different data and process definitions need to be integrated. Semantic annotation of all aspects of Web service, including their capabilities, data and processes, is an important step towards making SOA a success. To fully make use of Semantic Web services(SWS), execution environments are required for semantics-enriched service discovery, selection, composition and invocation. A specification for such an environment is the objective of the OASIS Semantic Execution Environment (SEE)[2]. This paper reviews the status of SOA and Semantic Web Services in Section 2 and looks at how SOA, the Event-Driven Architecture (EDA) [7] and existing SWS research relate to the developing specification of SEE. Section 3 briefly describes the state of the OASIS SEE and Section 4 concludes the paper.

2 Background

The main rationale underpinning SOA is the conceptualization of IT systems' capabilities as well-defined, independent, invocable, distributable and typically

[1] Funded by EU Integrated Projects, DIP FP6-507483 and SUPER FP6-026850
[2] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=semantic-ex

coarse-grained services [5]. Conceptually, SOA prescribes that service specification be decoupled from implementation to achieve what is regarded as the main technical benefit of SOA, i.e. maintaining a loose coupling between integrated components. Despite their success, there still remain important challenges to be addressed in current SOA-based solutions. Discovering and composing suitable services are typically part of any development process based on SOA but, nothing is prescribed for effectively supporting these activities. UDDI is the most well-known specification for an XML-based registry of service descriptions on the Web but the descriptions are *syntactic* only - the meaning is still open to interpretation by the user. Similarly, service composition is mainly based on the syntactic descriptions provided by WSDL, which are necessary but not sufficient, since the semantics remain implicit and cannot be automatically processed [3].

Additional architectural inconveniences arise from the fact that in typical SOA implementations, clients establish one-to-one synchronous communications with services. Firstly, synchronous communications are not always suitable or even possible as they may lead to reliability, latency and quality-of-service degradation. Secondly, there remains the need for further decoupling since clients initiate services execution and they must therefore know the services to be used in advance. Thus, dropping a new service into an existing SOA system does not imply that existing software will directly start taking benefit from this new capability offered. Facing these challenges, major IT vendors are currently proposing coupling SOA with EDA [7]. EDA is another architectural style that prescribes that communication between components has to be performed on the basis of event notifications, where events are basically understood as changes in the state of something relevant for the system. Communication between system components solely takes place through event notifications which are generated by so-called *event producers*. These events are populated by the Message-Oriented Middleware, and may trigger reactions by so-called *event consumers*. In contrast with SOA, event producers and event consumers are completely decoupled and many-to-many asynchronous communications between them are supported, the communication middleware being in charge of appropriately distributing over the systems components.

Coupling SOA and EDA we obtain a particularly appealing architecture for supporting the effective execution of possibly distributed and completely decoupled services, synchronously or asynchronously executed in an efficient manner. Still, as can be distilled from the state-of-art, this is not all there needs to be [7]. Event-based communication is indeed a useful way for a decoupled integration but it does not, and it actually cannot, avoid the need to agree on the events and the need for mediating between heterogeneous data models [7]. Designing new systems will require establishing an agreement on which events there need to be, their syntax and their actual semantics. Thus, adapting or, as we prefer to call it, *mediating* between data models remains a critical requirement.

In parallel with the work performed in the Software Architecture community, research on Semantic Web Services has been devoted to supporting a greater automation of the discovery and composition of Web Services. This research mainly

builds upon Semantic Web technologies in order to deal with data heterogeneity. The METEOR-S [9] project aims to extend existing technology for annotating, discovering, composing and executing Web services with semantics. Central to METEOR-S is the combination of WSDL-S [1], a bottom-up approach to semantic annotation of Web services, with the use of planning techniques. OWL-S [6] defines an ontology for Semantic Web Services based on the Web Ontology Language (OWL). Multiple tools are available but they are mostly independently developed and not part of a cohesive architecture. Finally, the Web Services Execution Environment (WSMX) [4], and the IRS-III [2] are initiatives based on the Web Services Modeling Ontology (WSMO) [8]. Both WSMX and IRS-III represent the main technical contributions for the Semantic Execution Environment we present next.

3 The Semantic Execution Environment

The semantic annotation of autonomous heterogeneous Web services is motivated by the drive to automate the discovery, composition and invocation of services across the Web. Realising this requires a supporting infrastructure of software that can interpret and, where appropriate, execute the semantic descriptions. As can be distilled from the previous discussion, a fully-fledged infrastructure that applies Semantic Web Services technologies and meets the strong Software Engineering requirements of real-life scenarios remains to be developed. To that end, the OASIS SEE technical committee is chartered to produce a guideline specification for a SWS execution environment.

SEE aims to reconcile state-of-the-art research on Semantic Web Services with mainstream architectural support for service-oriented systems in what we refer to as a Semantic Event-Based Service-Oriented Architecture. SEE specifies a generic architecture for the Goal-based discovery, composition, mediation and invocation of Web services based on their semantic annotation using the conceptual model provided by WSMO.

A fundamental principle of SEE is the decoupling of its components. These do not invoke each other directly either via local or remote object methods. Instead SEE adopts an event-based mechanism where components subscribe to events that they can consume and can publish events intended for other components. From the design perspective, there is no constraint on components being located on different physical servers as long as a network connection is available.

Analogously to semantic annotation of data and services, reaching full semantic interoperability between SEE components requires a formalisation of the events being exchanged. We initially identify two categories of events - data and process. Data events refer to a business occurrence relevant for the system to achieve its goals e.g. discovery request, mediation request etc. Formalising data events is, as we previously discussed, necessary for achieving a convenient framework for interoperability. In contrast, process events relate to the state of activities e.g. *start of discovery* and aim to support workflow monitoring and compensation.

4 Conclusions

In this paper, we highlight the symbiosis between the SOA and EDA architectural styles for supporting flexible service-oriented systems . We have introduced the main principles underlying SEE, which is based on what we refer to as Semantic Event-Based Service-Oriented Architecture, a novel architectural style that aims to circumvent the drawbacks of SOA and EDA by providing formal semantics for all aspects of the architecture. Our analysis highlights the need for applying semantics to various aspects of the infrastructure to support the Semantic Web services vision. Additionally we have identified the need for formalizing what we refer to as *Data* and *Process* events, in order to support modifying, validating and monitoring the execution environment. Future work in the context of SEE will be devoted to formalizing the components, formalizing Data and Process Events, and applying workflow mining and monitoring techniques for the validation and monitoring of execution environment.

References

1. R. Akkiraju, J. Farrell, J.Miller, M. Nagarajan, M. Schmidt, Amit. Sheth, and Kunal Verma. Web service semantics - wsdl-s, available at http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/. Technical report, 2005.
2. Liliana Cabral, John Domingue, Stefania Galizia, Alessio Gugliotta, Barry Norton, Vlad Tanasescu, and Carlos Pedrinaci. IRS-III: A Broker for Semantic Web Services based Applications. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, Athens, GA, USA, 2006. (To appear).
3. J. Euzenat. Towards a principled approach to semantic interoperability. In *Proceedings of IJCAI workshop on Ontologies and information sharing*, Seattle (WA), US, 2001.
4. Armin Haller, Emilia Cimpian, Adrian Mocan, Eyal Oren, and Christoph Bussler. WSMX – A Semantic Service-Oriented Architecture. In *Proc. of the 3rd Int. Conf. on Web Services*, pages 321 – 328. IEEE Computer Society, 2005.
5. C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz. OASIS Reference Model for Service Oriented Architecture V 1.0. Technical report, OASIS, July 2006.
6. D. Martin and et. al. OWL-S: Semantic Markup for Web Services, version 1.1 available at http://www.daml.org/services/owl-s/1.1/. Technical report, 2004.
7. G. Mühl, L. Fiege, and P. Pietzuch. *Distributed Event-Based Systems*. Springer-Verlag, 2006.
8. Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Ruben Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Christoph Bussler, and Dieter Fensel. Web Service Modeling Ontology. *Applied Ontology*, 1(1):77–106, 2005.
9. Kunal Verma, Karthik Gomadam, Amit P. Sheth, John A. Miller, and Zixin Wu. The METEOR-S Approach for Configuring and Executing Dynamic Web Processes, Technical Report, available at http://lsdis.cs.uga.edu/projects/meteor-s/techRep6-24-05.pdf. Technical report, 2005.