Analyzing Orchestration of BPEL Specified Services with Model Checking

Joseph C. Okika*

Supervised by Prof. Anders P. Ravn
Computer Science Dept. Aalborg University, Aalborg, Denmark

Abstract. This project investigates, implements and evaluates tool support for analysis of SOA-Based service contracts using Model Checking. The specification language for the contract is Business Process Execution Language (BPEL). It captures the behavior of services and allows developers to compose services without dependence on any particular implementation technology. A behavior specification is extracted from a BPEL program for formal analysis. One of the key conditions is that it reflects the intended semantics for BPEL, and in order to make it comprehensible, it is specified in a functional language. The resulting tool suite is hosted on an Eclipse platform.

1 Research Question and Its Significance

Service Oriented Architectures (SOAs) are applicable when multiple applications running on varied technologies and platforms need to communicate with each other. In this way, enterprises can mix and match services to perform business transactions with less programming effort. However, a service operates under a contract/agreement which will set expectations, and a particular ontological standpoint that influences its semantics [14]. Services are first class citizens and are autonomous as well as distributed in nature. They can be composed to form higher level services or applications to solve business goals. Of course, this raises a lot of issues such as managing composed services, monitoring their interaction, analyzing the behavior of interacting services, verifying the functionality of individual services as well as composed services.

So far, service development has used traditional testing which are inefficient when dealing with distributed systems. Thus, there is a clear need to employ and integrate successful analysis techniques like model checking in the design of support tools for effectively solving these problems as well as in the implementation of high quality SOA-based services. Therefore, a detailed contractual description of services and corresponding semantics is of great importance.

BPEL offers a programming model for specifying the orchestration of web services through several activities. Activities are categorized into two; basic and

* Partially supported by the Nordunet3 project COSoDIS - "Contract-Oriented Software Development for Internet Services". E-mail: ojc@cs.aau.dk

structured. Basic activities (for instance invoke, receive, etc.) define the interaction capabilities of BPEL processes whereas the structured activities are made up of constructs such as flow (for synchronization), compensate, and pick among other activities.

Current Results In [4] we have demonstrated a viable solution to the problem of checking for some functional and behavioural properties of individual services. This is done through translation of the specifications to timed automata followed by model checking for relevant properties. In [2] we consider the problem of consistency across specifications and identified a need to set up a correspondence between the individual automata. The novel contribution in that paper is to make such a consistency check practical by translating the automata to CCS, the input language for the Concurrency Work Bench. As demonstrated by a case study, this technique is applicable and gives a handle for automating yet another consistency check for web services.

2 Current knowledge and the existing solutions

In this section, we present several efforts geared toward formalizing/analyzing services specified using BPEL - one of the most widely used orchestration language. The overall observation about these works is that they all deal with three major issues; semantics definition, mapping to an analysis language and applicability. In Figure 1, δ represents those efforts that cover semantics definition and mostly applying Petri net simulation while μ represents those that focus on mapping/translation to an analysis language.

The issue with most of these results is that they cover only fragments of the language and for some of them, there is not explicit statement about the underlying analysis language and possibility of automation.

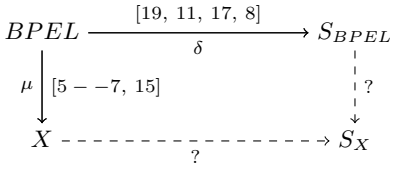

Fig. 1. BPEL Formalizations

Abstract state machines are used in [16] to define an abstract operational semantics for BPEL for version 1.1. The work focuses on formal verification of service implementations and resolving inconsistencies in the standard. Abouzaid and Mullins [1] propose a BPEL-based semantics for a new specification language based on the π-calculus, which will serve as a reverse mapping to the π-calculus based semantics introduced by Lucchi and Mazzara [10]. Their mapping is implemented in a tool integrating the toolkit HAL and generating BPEL

code from a specification given in the BP-calculus. Unlike in our approach, this work covers the verification of BPEL specifications through the mappings while the consistency of the new language and the generated BPEL code is yet to be considered. As a future work, the authors plan to investigate a two way mapping.

Several model checking approaches have been employed to provide some form of analysis. An overview of most of the semantics foundation is given in [18]. An illustrative example which is well-explained is [11]. It deals with specification of both the abstract model and executable model of BPEL. The approach is based on Petri nets where a communication graph is generated representing a process's external visible behavior. It verifies the simulation between concrete and abstract behavior by comparing the corresponding communication graphs. Continuing with Petri net, an algebraic high-level Petri net semantics of BPEL is presented in [17]. The idea here is to use the Petri patterns of BPEL activities in model checking certain properties of BPEL process descriptions. The model is feature complete for BPEL 1.1. Lohmann extends this work with a feature-complete Petri net semantics for BPEL 2.0 [8].

As there exists several BPEL formalizations including a comprehensive and rigorously defined mapping of BPEL constructs onto Petri net structures presented in [19, 13] a detailed comparison and evaluation of Petri Net semantics for BPEL is presented in [9]. The comparison reveals different modelling decisions with a discussion on their consequences together with an overview of the different properties that can be verified on the resulting models.

In the case of using labeled transition systems as models for formalizing BPEL, few efforts is found in the literature which focuses on some fragments of BPEL constructs. For instance, Geguang et al. present a language μ-BPEL [15] where a full operational semantics using a labeled transition system is defined for this language and its constructs to Extended Timed Automata. The language constructs are mapped to a simplified version of BPEL 1.1. Fu et al. presented a translation from BPEL to guarded automata in their work [5]; the guarded automata is further translated into Promela specification which is the language for the SPIN model. Similar approaches are also followed in [6, 7]. All these efforts points to the fact that there is an important need for service contracts to be specified and analyzed.

3 Proposed Ideas

As mentioned in the previous section, many theoretical results have considered the semantics analysis of BPEL. However, there are several issues around these semantics. First, there is the issue of coverage - that is to say, is the full BPEL language covered or some fragments of it? Most of the efforts using automata as presented in the previous section covers only some fragments of BPEL. A few of the efforts using Petri net covers a feature-complete BPEL. We point this out because it is worthwhile to have a comparison with another full coverage in a different formalism like automata. Second, there is the issue of translation where

one may ask: is it semantic preserving? There is also the third issue of whether it is manual, semi-automated or automated.

Figure 2 shows the proposed approach; mapping BPEL to timed automata (TA). This defines a semantics for BPEL with a clear description of what is included and what is abstracted in the mapping and thus answers the issues raised above.

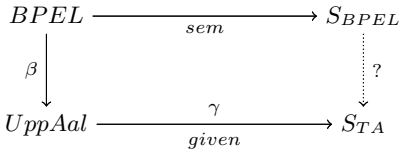

Fig. 2. The new Approach

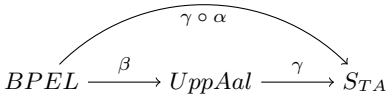

Fig. 3. Functional Composition

Looking at Figure 2, starting from BPEL, we consider a full behavior of BPEL syntax and define the semantics based on UppAal, S_{BPEL}. We follow a functional approach where we define a function β mapping BPEL to UppAal. We use timed automata for the formal model but with a rendering to UppAal because it is a mature model checking tool with wider audience and supported in our research environment. It can be a different choice (for example SPIN) in another environment. Note that the function *given* which takes care of the TA semantics is given with the UppAal tool and it's transition system semantics. Composing these two functions as shown in Figure 3 relates $BPEL$ to S_{TA}. In effect, having defined the function mapping BPEL to UppAal, we achieve a semantic preserving extraction/translation. That is, taking the inverse of the function gives us the result.

In [12] we give a classification of service contract specification languages based on application families and aspects. The classification identifies competing languages across aspects. It shows where a language may fit into the development of service based applications as well as the ones that allow for desired analyses, for instance match of functionality, protocol compatibility or performance match. In addition, we use the classification to survey analysis approaches. Furthermore, the classification may assist in planning of development activities, where an application involves services with contracts that span across families. Such scenarios are to be expected as service oriented applications spread. Another paper [3] focuses on analyzing behavioral properties for web service contracts formulated in Business Process Execution Language (BPEL) and Choreography Description Language (CDL). The key result reported is an automated technique to check consistency between protocol aspects of the contracts. The contracts are abstracted to (timed) automata and from there a simulation is set up, which is checked using automated tools for analyzing networks of finite state processes.

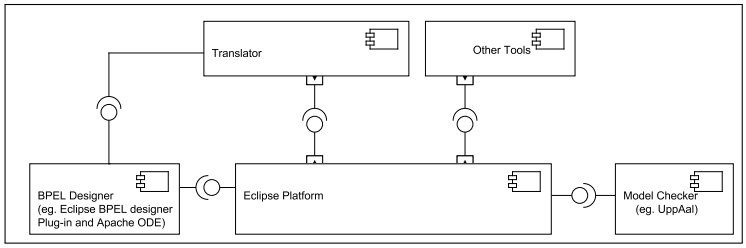

Fig. 4. Plug-in Architecture

4 Contribution to the problem domain and Discussion

The project offers three distinct contributions in the development of analysis and verification tools for SOA-based services. 1) The technique employed is a rigorous use of the power of functional languages in defining a property preserving mapping for the full behavior of BPEL. 2) Model checking of behavior properties of BPEL. General properties such as those related to deadlock and reachability as well as application specific properties are considered. Eg., services should not deadlock even with faults and compensation. 3) A prototype Integrated tool. The supporting tool will allow developers to leverage the already existing IDE such as Eclipse to design, specify and analyze SOA-based services.

Tool Development: We focus on building a theory based tool that gives developers of SOA-based services a clear understanding of BPEL processes. We are implementing the integrated supporting tool as a plug-in in the Eclipse framework. A model (UML) of the various components of the analysis tool is shown in Figure 4.

Discussion: The main novelty is to solve the issue of semantics unrelated to analysis tools. This is achieved by defining the extraction function using a functional language. As a side effect to this, we develop a functional XML parser/unparser for Standard ML. As this is an ongoing work, further effort will be geared toward tuning the tool. We plan to build a service based point of sale system using ActiveVOS orchestration system to demonstrate analysis of properties.

References

1. Faisal Abouzaid and John Mullins. A Calculus for Generation, Verification and Refinement of BPEL Specifications. *Electron. Notes Theor. Comput. Sci.*, 200(3):43–65, 2008.
2. Emilia Cambronero, Joseph C. Okika, and Anders P. Ravn. Analyzing Web Service Contracts - An Aspect Oriented Approach. In *Proceedings of the International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM'2007)*, pages 149 – 154. IEEE Computer Society Press, November 2007.

3. Emilia Cambronero, Joseph C. Okika, and Anders P. Ravn. Consistency Checking of Web Service Contracts. *Int'l Journal On Advances in Systems and Measurements*, 1(1):29–39, 2008.
4. G. Diaz, J. J. Pardo, M. E. Cambronero, V. Valero, and F. Cuartero. Verification of Web Services with Timed Automata. In *Proceedings of First International Workshop on Automated Specification and Verification of Web Sites*, volume 157, pages 19–34. Springer Verlags Electronics Notes Theor. Computer Sci. series, 2005.
5. Xiang Fu, Tevfik Bultan, and Jianwen Su. Analysis of interacting BPEL web services. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 621–630, New York, NY, USA, 2004. ACM.
6. Xiang Fu, Tevfik Bultan, and Jianwen Su. WSAT: A Tool for Formal Analysis of Web Services. In *Proc. of 16th Int. Conf. on Computer Aided Verification*, pages 510–514. Springer, 2004.
7. Xiang Fu, Tevfik Bultan, and Jianwen Su. Synchronizability of conversations among web services. *IEEE Transactions on Software Engineering*, 31(12):1042–1055, December 2005.
8. Niels Lohmann. A feature-complete Petri net semantics for WS-BPEL 2.0. In Kees van Hee, Wolfgang Reisig, and Karsten Wolf, editors, *Proceedings of the Workshop on Formal Approaches to Business Processes and Web Services (FABPWS'07)*, pages 21–35. University of Podlasie, June 2007.
9. Niels Lohmann, H.M.W. Verbeek, Chun Ouyang, and Christian Stahl. Comparing and evaluating Petri net semantics for BPEL. *IJBPIM*, 2008. (Accepted for publication).
10. Roberto Lucchi and Manuel Mazzara. A pi-calculus based semantics for WS-BPEL. *J. Log. Algebr. Program.*, 70(1):96–118, 2007.
11. Axel Martens. Consistency between executable and abstract processes. In *EEE '05: Proceedings of the 2005 IEEE International Conference on e- Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service*, pages 60–67, Washington, DC, USA, 2005. IEEE Computer Society.
12. Joseph C. Okika and Anders P. Ravn. Classification of SOA Contract Specification Languages. *Web Services, IEEE International Conference on*, 0:433–440, 2008.
13. Chun Ouyang, Eric Verbeek, Wil M. P. van der Aalst, Stephan Breutel, Marlon Dumas, and Arthur H. M. ter Hofstede. Formal semantics and analysis of control flow in WS-BPEL. *Sci. Comput. Program.*, 67(2-3):162–198, 2007.
14. R. Perrey and M. Lycett. Service-oriented architecture. *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on*, pages 116–119, 2003.
15. Geguang Pu, Xiangpeng Zhao, Shuling Wang, and Zongyan Qiu. Towards the Semantics and Verification of BPEL4WS. *Electr. Notes Theor. Comput. Sci.*, 151(2):33–52, 2006.
16. D. Fahland W. Reisig. ASM-based semantics for BPEL: The negative Control Flow. In *Proc. 12th International Workshop on Abstract State Machines*, pages 131–151, 2005.
17. Christian Stahl. A Petri Net Semantics for BPEL. Informatik-Berichte 188, Humboldt-Universitt zu Berlin, July 2005.
18. Frank van Breugel and Maria Koshika. Models and Verification of BPEL, 2005. http://www.cse.yorku.ca/ franck/research/drafts/tutorial.pdf.
19. M. T. Wynn, H. M. W. Verbeek, Aalst, Ter A. H. M. Hofstede, and D. Edmond. Business process verification - finally a reality! *Business Process Mgmt. Journal*, 15(1):74–92.