Practical Web Service Composition with Parametric Design

James Scicluna

STI Innsbruck, Austria
james.scicluna@sti2.at

Abstract. Web Service composition is one of the central topics in service oriented architectures. While industry heavily relies on static composition - through the use of process modelers and execution languages - academia focus more on the dynamic aspect - using search techniques to look for a solution. In this work, we attempt to leverage between the static and dynamic approaches using a Parametric Design methodology. Templates of service bundles define the skeleton and parameters and constraints within the bundle define the requirements over the needed services. We do this work in the context of the seekda service platform.

1 Introduction

In this work, we will tackle problems around web service composition, an open research topic in the realm of service oriented architectures (SOA). Service composition takes the nature of static, dynamic and at times hybrid. The first is more popular in industry settings; the latter two are still a popular open topic of research in academia. Static composition demands substantial human effort and is time consuming, and thus is undesirable in a dynamic environment such as that of an SOA. On the other hand, dynamic composition is mostly done in an automatic fashion. This is a hard problem in general, often requiring a relaxation of the problem or ignoring a subset of the constraints. We will leverage between the static and dynamic approaches and in particular focus on starting with a composition/process template and iteratively refine it until a complete solution is reached. We call this solution a service bundle. This process will be performed using a Parametric Design methodology[5] which - with adequate search methods that we will develop - iteratively refines a template until all parameters and constraints are fulfilled. Our aim is to make the service composition as practical as possible in an industry setting. We will use the seekda[1] service platform to test our solutions.

In the rest of the document, we will first describe the current state of the art in service composition - Section 2 - and proceed with an outline of our solution and approach to the problem - Section 3. Section 4 concludes.

[1] http://seekda.com/

2 State of the Art

We now review both static and dynamic approaches for service composition. We start with notations and tools followed by executable composition languages. The most popular notation to model processes is claimed to be BPMN[2], a graphical representation allowing creating business processes in a workflow-style. It supports business process management for both business and technical users. Another popular notation is EPC which is a graph-based approach - in contrast to BPMN, which is block-based. EPC allows various connectors for alternative and parallel execution of processes and it uses logical operators such as OR, AND, and XOR. While BPMN is more intuitive, EPC is formally defined avoiding potential ambiguity problems which may arise in BPMN. These notations do not allow to express parameters and constraints required for composition to take place.

One of the most widely used tools for graphical representation of processes is Enterprise Architect. This tool provides a comprehensive platform to model software components and services using standard UML. This is a more generic tool and not directly focused on business processes. Furthermore, it does not integrate any composition approach or annotations for the processes. Maestro for BPMN [1] (from SAP Research) integrates semantic technologies with an industry-level application for annotating processes and composing services. However, it employs a fully automatic composer which generally results in a sequence of web services that need to be re-arranged by a process engineer. Another research effort for semantically modelling processes is WSMO-Studio, in particular the Business Process Modelling Ontology (BPMO) Plugin. This plugin has been developed within the European Integrated Project, SUPER[3]. It allows to semantically annotate processes as well as discovering goal tasks using a composition tool developed with the same project [2]. This is an open source project and purely academic and its current status (together with the semantic technologies it employs) are currently infeasible to be used within an industrial productive setting.

BPEL4WS [6] is an executable process language proposed by IBM and Microsoft. It allows creating abstract and concrete specifications. At the abstract level, a process is described in terms of entities that participate in the interaction (through the use of roles). At the concrete level, bindings to actual services are defined in order to provide the actual functionality required during the execution (usually through WSDL). BPEL4WS is claimed to be the most widely used execution language within industry. Another approach emerging from the World Wide Web Consortium (W3C) is WS-CDL[10]. This language allows specifying interactions from a global perspective, rather than from the view of a single entity (the approach taken by BPEL). This language is not however yet so widely used. We envision such an executable language to be the result of a fulfilled

[2] http://www.bpmn.or
[3] http://www.ip-super.org

service bundle, ready for execution. In this sense, we do not go beyond the state of the art but simply adapt existing technologies.

What follows are dynamic approaches for service composition. As mentioned earlier, one possibility is to start from a skeleton (or template) of a process and constantly refine it until a solution is reached. The refinement is performed using some problem solving technique such as Extend-Model-then-Revise (EMR), Complete-Model-then-Revise (CMR), Hill Climbing (CMR-HC) and CMR-A*[5]. To best of our knowledge, there is just one piece of work that directly uses Parametric Design for the Web Services composition[9]. In this work, a complex Web service is described as a template that must be configured for some specific use. The problem solving method they use is called Propose-Critique-Modify. The solution template is based on the OWL-S but the actual broker that refines the solution template is implemented in Prolog. Although an adequate configuration is reached, the broker requires a substantial amount of high quality knowledge which may not be available in the typical Web Services scenarios. Also, candidate configurations are tested by executing them which is not desirable in real world settings where the effects of execution are sometimes irreversible.

Similar ideas to the concept of template refinement can also be found in [4, 3]. In [4], Golog is augmented, combining online execution of information-providing Web Services with offline simulation of those that alter the real world. This is achieved through the implementation of generic procedures. In [3], abstract BPEL4WS descriptions are semantically translated (hence, bottom-up approach). Reasoning is performed over these semantic descriptions such that adequate Web services are discovered. These approaches are quite generic and generally inefficient, thus making them infeasible for practical service composition..

Perhaps, the most widely spread approaches towards composition are those that employ A.I. Planning techniques Traditional approaches attempt to search for a solution by constructing the composite process from scratch (typically using an Artificial Intelligence Planning approach)[4, 7, 8]. The result is a sequence of Web Services that achieve the desired goal. However, such a solution would be in many cases inadequate: fault-handling is ignored due to the sequential nature of the solution and due to the fact that dealing with this aspect is hard in general; names of inputs/outputs must precisely match and thus solutions that may actually exist (e.g., through a subsumption relation between parameters) are not found; there is no notion of parallel execution. Some of these problems have been tackled [2] by a trade-off between expressivity and scalability. Still, in most practical scenarios, the resulting solution is not adequate and thus requires a Process Designer to remodel the process as needed.

3 Proposed Solution and Approach

As stated, our objective is to leverage between static and dynamic composition. Concretely, our objectives are the following:

– adapt (and possibly further develop) an existing semantic language that is expressive enough to describe constraints in process templates but simple enough to not hamper efficiency of the composition
– extend an existing graphical notation for describing processes, accommodating the developed parameter language
– design efficient refinement algorithms for fulfilling parameters and constraints of the process templates, taking particular care to scalability; [2] is of particular importance in this respect
– evaluate these methods over different scenarios

In order to achieve these targets, we will develop a Bundle Configurator & Assistant tool that enables the creation of templates, editing of parameters and constraints and assists in the fulfilment of the template using Parametric Design. The development of this tool will proceed in the order of our mentioned targets. To illustrate our vision of the templates and the generation of service bundles, we describe a hotel brokering scenario (Figure 1). The example shows a typical process template - the starting point for the composition - followed by the resulting service bundle.

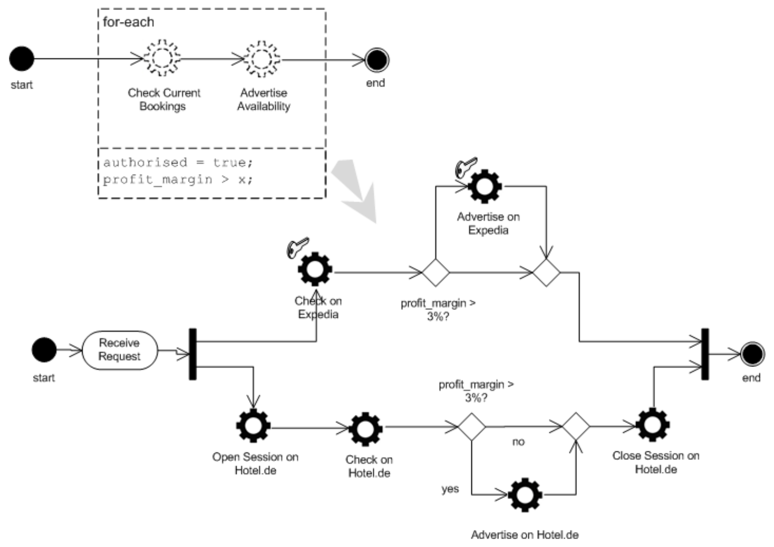

Fig. 1. An example of a template refinement process.

At the template level, the process is very simple. A Business Analyst defines the process in its primitive form, the start and end nodes and the checking and advertise tasks. These tasks provide a very high level description of what the process must do. Additional constraints are specified, for example, a Technical Process Designer specifies that these actions must be performed in an authorised way. A Marketing Manager further specifies that the profit-margin obtained from advertising on a channel must be greater than some value. The generated process defines first a "Receive" action which receives a request from the Hotelier to start updating the respective channels. The process deals with two channels which are updated in a parallel fashion. There is a considerable difference in the behaviour of each channel. For Expedia, credentials must be sent each time there's communication taking place, but this is not the case for hotel.de, in which opening a session would be just enough. In any case, both sub-processes communicate in an authorised manner, as defined by the initial constraint of the bundle template. In both sub-processes, the profit margin constraint is dealt with a branching control flow.

4 Conclusion

Although this work is in its early stages, we believe that Parametric Design methodologies are very adequate to address service composition in an effective way. We shall investigate the usage of appropriate parameter languages and afterwards develop efficient algorithms for fulfilling the constraints specified by such a language. Finally, we will evaluate our results on a real industry setting.

References

1. M. Börn, J. Hoffmann, T. Kaczmarek, M. Kowalkiewicz, I. Markovic, J. Scicluna, I. Weber, and X. Zhou. Semantic annotation and composition of business processes with maestro. In *Proceedings of the 5th European Semantic Web Conference (ESWC'08)*, 2008.
2. J. Hoffmann, I. Weber, J. Scicluna, T. Kaczmarek, and A. Ankolekar. Combining scalability and expressivity in the automatic composition of semantic web services. In *Proceedings of the 8th International Conference on Web Engineering (ICWE'08)*, Yorktown Heights, USA, July 2008.
3. D. Mandell and S. McIlraith. Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. In *Proc. of ISWC'03*, 2003.
4. S. McIlraith and T. Cao Son. Adapting Golog for composition of semantic Web services. In *Proc. of the 8th Int. Conf. on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France*, 2002.
5. E. Motta and Z. Zdrahal. Parametric design problem solving. Technical report, The Open University, XXXX.
6. OASIS. Web Services Business Process Execution Language, August 2006.
7. M. Pistore, P. Traverso, and P. Bertoli. Automated Composition of Web Services by Planning in Asynchronous Domains. In *Proc. ICAPS'05*, 2005.
8. K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated discovery, interaction and composition of semantic web services. *Journal of Web Semantics*, 1(1):27–46, 2003.
9. A. ten Teije, F. van Harmelen, and B. Wielinga. Configuration of web services as parametric design. In *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management*, 2004.
10. W3C. Web Services Choreography Description Language, Version 1.0, 2005. W3C Recommended Draft 9 November 2005.