Automatic Service Composition for the Internet of Services

Erwin Aitenbichler, Stephan Borgert, and Max Mühlhäuser
Telecooperation Lab, Technische Universität Darmstadt
Hochschulstrasse 10, 64289 Darmstadt, Germany
{erwin,stephan,max}@tk.informatik.tu-darmstadt.de

Abstract

Next generation SOA systems promise to enable an "Internet of Services" (IoS) - an open environment, in which every participant is free to offer and consume services. It is vital for businesses to ensure process compatibility and to have the ability to quickly adapt to changes in service offering. This raises the need for new service and process description methods providing a formal foundation and well-defined semantics. In this paper, we discuss some challenges building the IoS and sketch how automatic service composition is implemented in the Theseus/TEXO project.

1 Introduction

Service-oriented Architecture (SOA) is an architectural style that facilitates loose coupling of components, and consequently enables flexible selection and substitution of services. However, today's SOA systems are rather closed. They are only used within the boundaries of an enterprise, or sometimes within conglomerates of enterprises with long-standing cooperations.

In contrast, the *Internet of Services* gives businesses the opportunity to utilize an open service market. It seems natural that there will be multiple offers for services providing the same functionality. The services offered on the market will be different in many details, such as their quality and how their internal processes are realized. Consequently, the parties interacting in a B2B scenario must ensure that their processes are compatible with each other, e.g., if a specific message is sent, then it must be expected on the other side, and processes must eventually terminate.

Some of the research challenges that arise in the realization of such an IoS are discussed in the following.

2 Process Modeling

The mainstream process description languages used today lack a formal foundation and well-defined semantics (e.g., EPC, BPMN), or they are too low-level (e.g., BPEL). Such descriptions do not permit computers to reason about processes. Beside these technical aspects, BPM also suffers from several other problems, as current practice in BPM projects shows.

Lack of Process Governance: A first fundamental problem is that processes are not 'lived' as they have been designed and modeled. Practitioners report that the vast majority of decisions made in a business are still based on gut feel, intuition and experience. In addition, if processes had initially been modeled, then the corresponding models are often not kept up-to-date. This impedes the assurance of process quality, analysis of process efficiency, and process improvement. Alone the discovery of how a process actually works in a business can cause significant costs: BPM consultants claim that they spend around 40% of the project time finding out how processes in a business actually run.

One-shot Transformations: During the IT implementation of a process often *one-shot transformations* are made. E.g., the implementation starts with a business analyst creating a BPMN model from an EPC sketch of the process, emphasizing the business aspects. Next, a software engineer creates an executable BPEL process. Because BPMN (2.0) only partially has well-defined semantics, automatic transformations produce a "BPEL skeleton" at best, lacking several technical details. Consequently, engineers transform from abstract to more concrete models and add details to the model (Figure 1). The relationships between model elements from the concrete to the abstract model get lost and it is not possible to automatically update the abstract model containing the business perspective.

"Outlook Processes": Another common implemen-

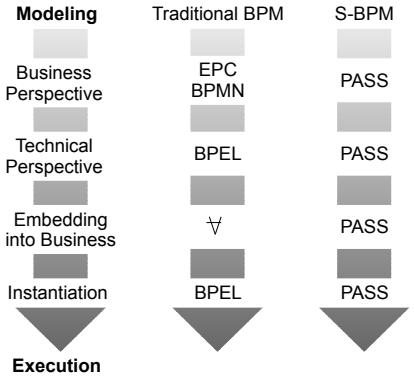

Figure 1: One-shot transformations in the traditional IT implementation of business processes vs. S-BPM.

tation of processes are so-called "Outlook processes". They contain activities like "send email to financial accounting", meaning that an employee uses ordinary email to perform a process step. This has two major drawbacks. First, the progress of process instances cannot be monitored directly, because the relationships between mails and process instances cannot be discovered easily. Second, because the email client allows unbounded communication, new communication paths can emerge easily. This changes the process, but is not reflected in the process model.

Consequently, we require process modeling languages to provide *formal semantics*, a *subject-oriented perspective*, *direct executability*, and *hierarchability* [1]. We currently use S-BPM (Subject-oriented Business Process Management) with the process description language PASS (Parallel Activities Specification Scheme) [2], which can be used throughout the whole process implementation, as PASS processes are directly executable as they are modeled. The concept of *embedding* allows to reduce the number of process variants, since the organizational structure of an enterprise can be modeled as a separate aspect apart from the process. PASS is based on the process calculus CCS. Such a formal foundation does not necessarily have a negative impact on usability: The jCOM1 Business Suite [3] provides easy-to-use process modeling tools.

3 Interaction Soundness

Process algebras provide well-studied algorithms for verification and behavioral equivalences. In addition, the CCS composition operator facilitates a hierarchization and modularization of the model, allowing to handle business processes of arbitrary size.

In a scenario where multiple different businesses cooperate, the processes internally performed by different services will typically also be modeled in different places. Because businesses are usually not willing to fully disclose their internal processes, it is necessary to derive the *behavioral interface* of a service. With the restriction operator, CCS provides the necessary means to perform this transformation automatically.

The overall process can then be composed by combining all known processes and behavioral interfaces of external services. In a valid composition, it must be ensured that the involved services are able to communicate properly with each other. Secondly, reachability analysis can be used to ensure that all end states of the subprocesses can be reached, where applicable.

4 Distributed Execution

PASS processes can be directly executed on a suitable execution engine. From the process model to the final execution of a process instance the following two additional steps are necessary: *embedding* and *instantiation*. Embedding maps subjects of the process to roles or services in the business. Instantiation denotes the creation of a new process instance for a concrete subject carrier.

When such processes are executed in a distributed system on multiple communicating engines, then the question arises how to correctly route messages from a process instance to remote process instances, while taking embedding information into account. Publish/Subscribe is a suitable abstraction for this application, because a communication abstraction is needed that supports dynamic endpoints and multicast.

Acknowledgments: This work was supported by the Theseus Programme, funded by the German Federal Ministry of Economy and Technology under the promotional reference 01MQ07012.

References

[1] Erwin Aitenbichler and Stephan Borgert. Application of Subject-oriented Modeling in Automatic Service Composition. In *S-BPM ONE*, CCIS 85. Springer, 2010.

[2] Albert Fleischmann et.al. Coherent Task Modeling and Execution Based on Subject-Oriented Representations. In *TAMODIA*, LNCS 5963, pages 78–91. Springer, 2009.

[3] jCOM1. S-BPM Suite. http://www.jcom1.com, 2010.