ReoService: Coordination Modeling Tool

Christian Koehler*, Alexander Lazovik**, and Farhad Arbab

CWI, Amsterdam, Netherlands
{koehler,a.lazovik,farhad.arbab}@cwi.nl

Coordination in SOA addresses dynamic topologies of interactions among services. Most efforts up to now have been focused on statically defined composition of services, e.g., using BPEL. To the best of our knowledge, there are no serious means to address the issues of dynamic coordination to accommodate continuously changing requirements. While BPEL is a powerful standard for service composition, it lacks support for typical coordination constraints, like synchronisation, mutual exclusion, and context-dependency.

In this paper we present ReoService, which is a modeling tool for coordinating business processes. ReoService is based on Reo [2] – a general framework for coordinating components in distributed systems. Reo is a channel-based exogenous coordination language wherein complex coordinators, called connectors, are compositionally built out of simpler ones. The simplest connectors are a set of user-defined communication channels with well-defined behavior. The emphasis in this model is on connectors, not on the services to connect. In this sense, ReoService acts as a "glue" language that interconnects and coordinates services in a distributed business process.

The Reo coordination tool is developed to aid the process designers who are interested in complex coordination scenarios. The ReoService and its underlying Reo framework are implemented in Java as a set of plug-ins [1] on top of the Eclipse platform (`www.eclipse.org`). Currently the framework consists of the following parts: (i) graphical editors, supporting the most common service and communication channel types; (ii) a simulation plug-in, that generates flash animated simulations on the fly; (iii) BPEL converter, that allows conversion of Reo connectors to BPEL and vice versa; (iv) Java code generation plug-in, as an alternative to BPEL, represents service coordination model as a set of Java classes; (v) validation plug-in, that performs model checking over coordinations represented as constraint automata.

We now describe the Reo framework architecture that is shown in Figure 1. The central part of the framework is a visual editor for Reo connectors. It represents the actual coordination model with services and communication channels. The developed tool also allows us to represent Reo in terms of constraint automata [4]–an alternative behavioral model. This is useful if additional validation based on model checking techniques [6] is required. Q-Automata [5] is used if

* The work in this paper is supported in part by a grant from the GLANCE funding program of the Dutch National Organization for Scientific Research (NWO), through projectWoMaLaPaDiA (600.643.000.06N09).

** This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

B. Krämer, K.-J. Lin, and P. Narasimhan (Eds.): ICSOC 2007, LNCS 4749, pp. 625–626, 2007.

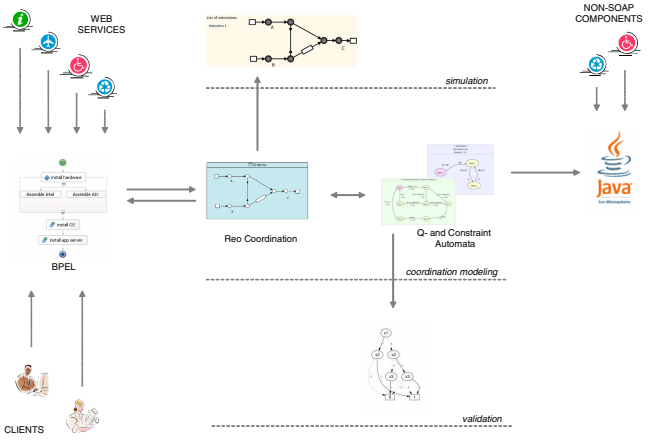

Fig. 1. Reo coordination framework for services

QoS aspects of communication channels are important. Along with editing, the Reo editor maintains simultaneous conversion to BPEL. To test the coordination model, one may run an animated simulation using the animation plug-in. In some situations it is desirable to use a coordination model in a non-web service scenario: in this case, generation of Java code is used. In the generated code non-SOAP components are represented by wrappers over Java threads.

However, our tools currently lack adequate support for certain concerns that are specifically important for services, e.g., preferences, extended service descriptions, and temporal constraints. While some of these concerns, e.g., temporal constraints, are naturally supported by our coordination language [3], others as extended service descriptions and preferences to provide users with better control over instantiated process execution require extensions that go beyond the scope of a general purpose coordination language. Improving our tools to support temporal aspects of Reo circuits is in our agenda. We also plan to address extended service descriptions and investigate preferences in our future work.

References

1. Eclipse coordination tools, `http://homepages.cwi.nl/~koehler/ect`
2. Arbab, F.: Reo: a channel-based coordination model for component composition. Math. Structures in CS 14(3), 329–366 (2004)
3. Arbab, F., Baier, C., de Boer, F., Rutten, J.: Models and temporal logics for timed component connectors (2004)
4. Baier, C., Sirjani, M., Arbab, F., Rutten, J.: Modeling component connectors in reo by constraint automata. Sci. Comput. Program. 61(2), 75–113 (2006)
5. Chothia, T., Kleijn, J.: Q-automata: Modelling the resource usage of concurrent components. In: FOCLASA 2006 (2006)
6. Klueppelholz, S., Baier, C.: Symbolic model checking for channel-based component connectors. In: FOCLASA'06 (2006)