The PiVizTool: Simulating Choreographies with Dynamic Binding

Anja Bog, Frank Puhlmann, and Mathias Weske

Business Process Technology Group
Hasso Plattner Institut for IT Systems Engineering
University of Potsdam
D-14482 Potsdam, Germany
{anja.bog,frank.puhlmann,mathias.weske}@hpi.uni-potsdam.de

Abstract. This paper presents a tool, the PiVizTool, for the interactive simulation of choreographies. Different participant instances can be added, deleted, and dynamically bound during the simulation. The PiVizTool uses the π-calculus as the formal foundation for representing dynamic binding. It provides an interactive graphical representation of complex choreographies that can be simulated according to different use cases.

1 Introduction

Nowadays, service-oriented architectures (SOA) are a common realization strategy for business process management (BPM). In a SOA, business processes are represented as services with internal and communication actions following a business logic. Multiple of these services work together in a choreography to provide a result. A core feature of service-oriented architectures is dynamic binding, which allows services to be bound at runtime.

Consider for instance figure 1, that shows the design-time view of a choreography in BPMN [1]. The initial participant is a customer, shown in the upper part. The customer places an order, a delivery address, and an invoice address at a reseller. The reseller has a number of manufacturers and payment organizations that handle each request. While at design-time the static interfaces of the manufacturer and the payment organization are given, different implementations might be bound during runtime. All of them directly bind to the addresses initially given by the customer to the reseller for providing their services.

2 The PiVizTool

The PiVizTool[1] provides an interactive environment for choreographies, where different participant instances can be added (imported) and selected during the

[1] The PiVizTool is available at `http://bpt.hpi.uni-potsdam.de/twiki/bin/view/Piworkflow/Simulator`.

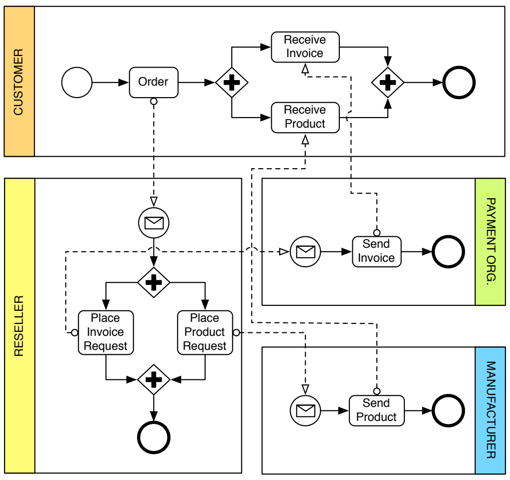

Fig. 1. A choreography in BPMN notation.

runtime of the simulation. The tool is grounded in the π-calculus [2], where the choreographies are formally defined according to [3,4]. The major difference between the π-calculus and other formalizations—such as Petri nets—is given by the inherent support of link passing mobility. Channels between business partners are denoted as links, where the links could be passed via other links (channels) to possible interaction partners. The technical foundation allows the direct representation of common patterns like asynchronous callback (problem here: correlate the matching process instances) as well as dynamic routing (problem: how to invoke interaction partners initially unknown).

An initial view of the given example in the PiVizTool is shown in figure 2. The main part shows the current state of the system representing the investigated choreography. Each participant of the BPMN diagram is depicted as a rectangle, representing an instance. All BPMN flow objects are shown as circles. The dependencies between the flow objects (sequence flows) are shown as lines with a dotted end marking the target. The message flows, however, have been restricted to the current knowledge of the system. We can see a link between the instances of the customer and the reseller, as well as two links from the reseller instance to a certain instance of the payment organization. There is no link back to the customer at this point in (simulation) time. The left hand side of the tool shows all dynamically created channels and links that are currently contained in the system. We can see `invAddr#21` and `itemAddr#20` representing the return channels of the invoice and the ordered items for the customer instance. The static channels, like `order` (shown in the main view between the customer and the reseller) are not contained in the list. Above the graphical view, the original as well as the current π-calculus representation of the system can be selected for viewing.

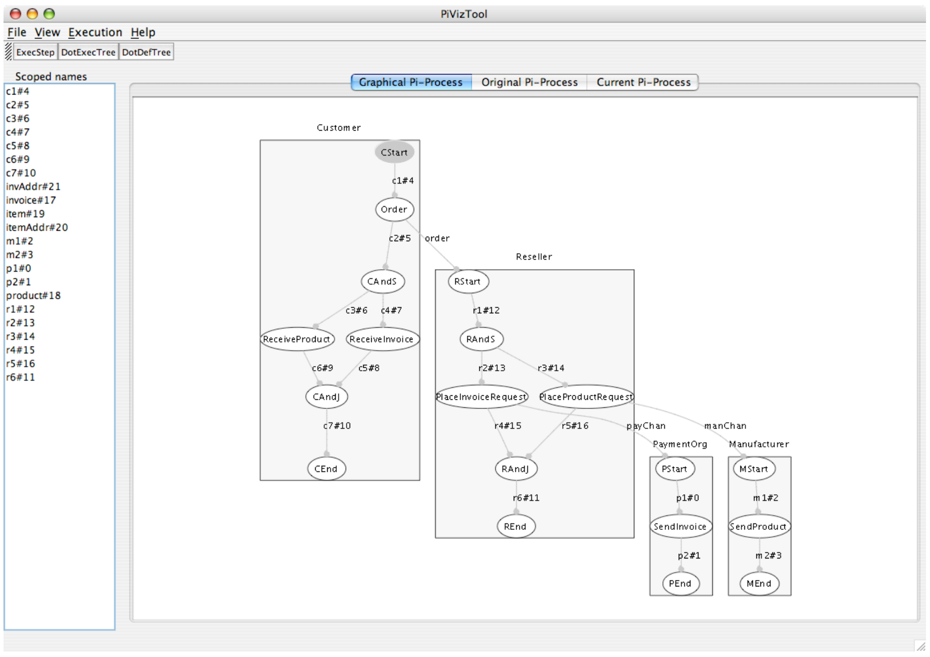

Fig. 2. The initial state of the example from figure 1 in the PiVizTool.

The user can execute an arbitrary action of the simulation—that is currently possible—by using the `ExecStep` button at the top of the interface. He can also click on an active element of the graphical representation to execute it. Active elements are either activities (depicted as circles filled with gray) or channels (depicted with bold, black lines). In the case of the example, the only action possible is the `CStart` activity that represents the start event of the customer. As stated, it can be triggered by clicking on it.

If an action is executed, the underlying π-calculus terms are evolved and a new graphical view of the system is created. The layout of the new view might be changed, because the visualization algorithm always calculates a view, where the interconnections between the nodes are minimized. Furthermore, new actions are made available according to the current state of the simulation.

After several actions, the state of the system has changed as shown in figure 3. The customer instance has placed the order, the delivery address, and the invoice address at the reseller. At this point in time, no channel between the instances of the customer and the reseller is existing. What has been added, however, is another instance of the payment organization via the `execution` menu of the PiVizTool that allows the addition of arbitrary participant instances during runtime (while not shown, it is even possible to instantiate completely different participants). Both instances of the payment organization can be reached via

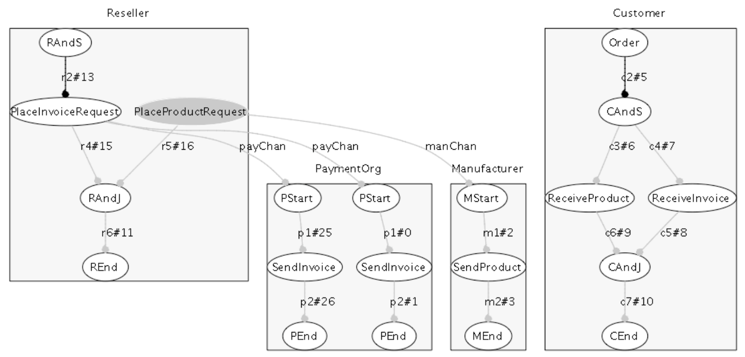

Fig. 3. The state of the example where the customer has placed his order and a second instance of the payment organization has been created.

the same channel, `payChan`. In the course of the simulation, the user can select the one he wants to activate.

3 Conclusion

This paper gave a short introduction to the PiVizTool that is able to simulate choreographies with dynamic binding based on the π-calculus. Currently, the input is restricted to plain π-calculus files, but we're working on enhancing the tool supported mapping of BPMN to pi-calculus based on our tool chain introduced in [5]. An extended description as well as the theoretical background is described in [6].

References

1. BPMI.org: Business Process Modeling Notation. 1.0 edn. (2004)
2. Milner, R., Parrow, J., Walker, D.: A Calculus of Mobile Processes, Part I/II. Information and Computation **100** (1992) 1–77
3. Puhlmann, F., Weske, M.: Using the Pi-Calculus for Formalizing Workflow Patterns. In: Business Process Management, volume 3649 of LNCS, Berlin, Springer Verlag (2005) 153–168
4. Overdick, H., Puhlmann, F., Weske, M.: Towards a Formal Model for Agile Service Discovery and Integration. In: Proceedings of the International Workshop on Dynamic Web Processes (DWP 2005). IBM technical report RC23822, Amsterdam (2005)
5. Puhlmann, F.: A Tool Chain for Lazy Soundness. In: Demo Session of the 4th International Conference on Business Process Management, CEUR Workshop Proceedings. Volume 203., Vienna (2006) 9–16
6. Bog, A.: A Visual Environment for the Simulation of Business Processes based on the Pi-Calculus. Master's thesis, Hasso-Plattner-Institute, Potsdam, Germany (2006)