A Toolkit for Choreographies of Services: Modeling, Enactment and Monitoring

Amira Ben Hamida, Julien Lesbegueries,
Nicolas Salatgé, and Jean-Pierre Lorré

Linagora R&D
3 Avenue Didier Daurat 31400 France
name.surname@linagora.com

Abstract. The support of the business community has considerably urged the advancement of the SOA by bringing useful supporting standards, for instance for Web Services description and collaboration design. Nevertheless, as the platforms are getting wider over geographically distant locations, there is a real need of keeping the link between the design time and the runtime. Model to model approaches ensure this kind of link, and in this context, choreography models help answering such issues. We bring our know how as ESB and BPM experts and propose an open source toolkit for services choreography. This toolkit provides a way to design a choreography, execute it on an Enterprise Service Bus and finally to monitor it. A Model to Model (M2M) top-down approach is implemented. We illustrate our purpose thanks to a business use case inspired from the CHOReOS European Project.

Keywords: Choreography, Service, SOA, EDA, Monitoring, ESB.

1 Motivations and Proposal

Large scale SOAs need complex design phases in order to be enacted. In particular, businesses interactions composing workflows executions are hot and risky points prone to failures. Moreover, they are hard to manage since the failure can come from one side, other side, or both ones. The OMG BPMN2.0 specification [1] answers to such issues by proposing a formal model to control these interactions: the Choreography conformance. We rely on a M2M transformation process that is composed of the following: (i) the choreography is designed thanks to BPMN2.0, (ii) a first transformation of the choreography specification leads to a set of executable processes, (iii) then a second transformation is operated from the choreography specification to a workflow monitoring model, and finally, (iv) finally, the choreography is enacted and monitored on the ESB. In the following, we present a standard-based approach implementing the aforementioned choreography-centric process. We provide a toolkit based on Business Process Modeling (BPM), and supporting functionality for choreography transformation

[1] http://www.bpmn.org/

P. Herrero et al. (Eds.): OTM 2012 Workshops, LNCS 7567, pp. 60–63, 2012.

and enactment on an Enterprise Service Bus (ESB), namely the Petals ESB. Furthermore, we implement an event-based monitoring based on Web Service Distributed Management [2].

2 Choreography Design

Very recently, the OMG BPMN2.0 specification rises as the de facto notation for modeling services collaborations and workflows. BPMN2.0 comes with new patterns facing the new challenges brought by the highly distributed and heterogeneous services. Indeed, it provides a means of modeling in a graphical and synthetic way complex collaborations between participants. Collaboration patterns such as the Choreography Tasks encapsulating the activity between 2 partners or Specific Gateways expressing the different possible paths in a large collaboration, are supported.

Most of the commonly used workflow designers either open source or proprietary such as Oryx [3], Intalio [4], Bonita [5], etc. start adopting the BPMN2.0 trend in their products. However, most of them do not provide a complete suite that handles the entire choreography lifecycle from its modeling to its deployment on a middleware. We argue that the choreography abstractions raise new challenges precisely in the M2M transformation, enactment and monitoring issues.

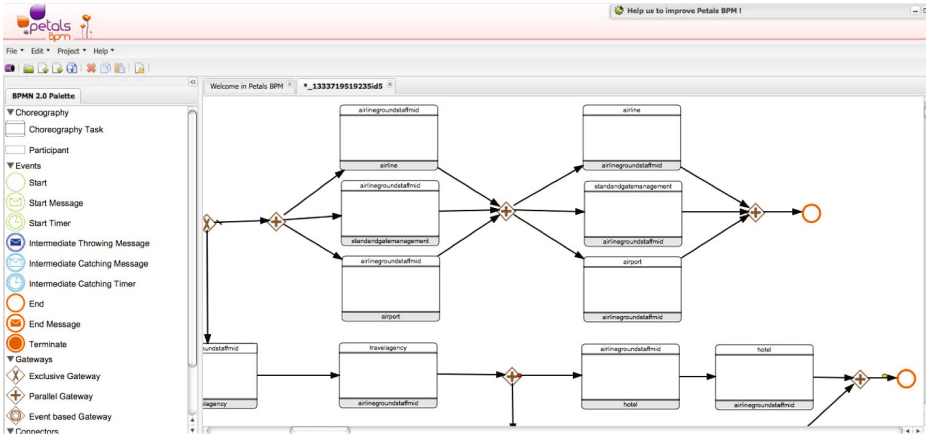

Fig. 1. Petals BPM Snapshot

The designed use case represents the case of managing an *unexpected arrival of a flight to an airport*. In this case, several participants take part to the scenario to ensure the passengers are transferred to the right places and hosted in

[2] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm
[3] http://bpt.hpi.uni-potsdam.de/Oryx/BPMN
[4] http://www.intalio.com/bpm/
[5] http://fr.bonitasoft.com/

hotels while waiting for the storm to pass. Thanks to the Petals BPM toolbar, a choreography designer can drag and drop the needed patterns to represent the participants collaborating. For instance, the airline ground staff contacts the travel agency to ask for handling the passengers arrival. Then, the airline ground staff needs to communicate to the airport the arrival of the flight and the troubles they are encountering. Each 2 communicating participants are expressed as doing a common activity into a Choreography Task. Though distinct, these participants take part in a wide choreography that acts for a final objective. In Figure 1, we show the aspect of this GWT Web tool as well as a snapshot of the scenario. We can see the Choreography Tasks involving each time 2 or more partners within a given activity, Inclusive and Exclusive Gateways showing the possible branches as well as the choreography Starting and Ending events.

The next section is dedicated to the transformation from the specification to the execution, and the enactment of the choreography over the middleware.

3 Choreography Deployment and Enactment

Once the choreography is designed, Petals BPM offers a mean to connect to a registry of services, and provides a matching utility to find services corresponding to the model tasks. When the choreography is fully concretized, the BPMN2BPEL transformation is executed. This transformation results in a set of orchestrations. Indeed, we consider that each BPMN pool corresponds to a business stakeholder and can be implemented as a WS-BPEL orchestration. Referring to our use case, the Choreography Task between the Travel Agency and the Airline Ground Staff, is concretized when each of these participants are satisfied by a service discovered in the registry. Once the Travel Agency and the Airline Ground Staff Roles are provided by services, then we proceed to the transformation from the specification to the execution. The resulted BPEL processes are deployed on the Petals ESB middleware. More precisely, the middleware is a set of ESB nodes deployed on the internet, distributed over the neighborhood of the different Web Services involved in the choreography. The deployment phase uses the Web Service Administration of the ESB that takes a BPEL process as input and stores it in an orchestrator (EasyBPEL engine) dedicated to the execution of the WS-BPEL 2.0-based processes. This top-down approach from specification to deployment allows managing the whole business process execution, in an automatic and controlled way. Furthermore, the choreography definition at design time enables the management of non-functional needs for monitoring the activity (See section 4).

4 Choreography Monitoring

We exploit our top-down approach for realizing an exhaustive choreography monitoring. More precisely, once deployed and enacted, a choreography is quite difficult to follow, by default, contrary to well structured orchestration processes. However, elements involved in it, such as services and orchestrations, are prone

to failures, and supervision needs to introspect them. Moreover, it is important for the supervision to know which partners of the choreography need to be warned, which ones are responsible for the failure, which parts can be replaced, etc. For that purpose, we implement a monitoring tool (EasierBSM), ensuring the monitoring of services, orchestrations and choreographies. Actually, the EasierBSM is a service bus, with a particular profile dedicated to monitoring. Its additional components provide an event-based mechanism based on the WS-BrokeredNotification[6]. Once an EasierBSM node subscribes to an ESB node, it receives its activity reports that are used by EasierBSM components to compute the 3 layers of monitoring. As Petals ESB, EasierBSM is a distributed bus that scales up dynamically by adding nodes to its topology. At the service level, specific components gather information about Quality of Service (QoS) and the Web Services SLAs [7] violations. This low-level monitoring allows the detection of the failing and non efficient services. At the orchestration level, processes status is given, allowing to know which branch of a process is executed. This allows to warn a particular stakeholder where its SOA application has failed. Finally, at the choreography level, a global workflow status is given, detailing which relationships have been successfully executed. In particular for this level, a top-down transformation is also operated. The original choreography model is passed to a component called EasierCOS, that compiles it and produces a monitoring model, executed in one or distributed way. This monitoring model is a graph onto which each node corresponds to a relationship of the choreography. Referring to our use case, the Choreography Task between the Travel Agency and Airline Ground Staff is expressed as a specific monitoring graph-based structure. The graph is updated as the choreography evolves and the services are invoking each other. If the Travel Agency stated in its SLA for a 3 minutes time response, then in case of any delay the Airline Ground Staff is notified. This way, we are able to detect within the whole choreography where is the failing task.

5 Conclusion

The toolkit we propose tries to face design issues related to large scale SOAs, and in particular, focuses on the monitoring part of choreographies they compose. Indeed, in addition of generating business execution of these SOAs, it provides a choreography monitoring engine able to detect stakeholders and operations involved in high level failures.

Acknowledgments. We present the result of the collective efforts deployed within our involvement in research projects. It was partially supported by the EC framework by the Play FP7-258659 and the CHOReOS FP7-257178 projects, and the French ANR project Soceda. The work reflects only the author's views. The community is not liable for any use that may be made of the information contained therein.

[6] `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn`
[7] `http://www.research.ibm.com/wsla/`