Monitoring Choreographed Services

L. Ardissono and R. Furnari and A. Goy and G. Petrone and M. Segnan
Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino, Italy

Abstract. **Web Service choreography management challenges the research on Service Oriented Architectures: different from Web Service composition, where a central orchestrator process invokes the Web Service suppliers, a choreographed service is decentralized and it is based on the coordination of a set of Web Service suppliers having a partial view of the overall service. In this paper, we present a framework which supports the monitoring of the progress of a choreographed service, the early detection of faults and the notification of the Web Services affected by the faults.**

I. Introduction

In the research on Service Oriented Architectures [9], the management of supply chains has been based on processes which orchestrate the execution of services within possibly complex workflows. For instance, the WS-BPEL language [7] is used to specify parameters and execution order of the invocations of operations on Web Service suppliers, as well as of the internal activities to be performed by the composite Web Service. Moreover, Abstract Processes, based on the WS-BPEL syntax, describe the invocation protocol of a conversationally complex Web Service supplier; this specification enables consumers to invoke the operations without violating the suppliers' business logics.

Web Service orchestration has been introduced to manage hierarchical supply chains in which a centralized entity invokes the suppliers. However, it has been recognized that, even in Enterprise Application Integration scenarios, the management of a non trivial supply chain often imposes the decentralized management of the composite service [5], e.g., in order to deal with the business requirements of suppliers which serve multiple consumers. In such situations, the cooperating Web Services have partial views of the overall service and they have to coordinate with each other in a complex multi-party conversation denoted as *choreography* [10]. The successful execution of the choreographed service thus depends on at least two factors: when the service is designed, Web Service suppliers have to be bound to each other (possibly by employing mediation services for interoperability purposes [6]); at service execution time, they have to perform operations successfully and in due time.

In this paper we propose a framework which supports the monitoring of the progress of a choreographed service, the early detection of faults and the notification of the Web Services affected by the faults. In our framework, a Monitor Web Service tracks the execution of the cooperating Web Services by analyzing their conversational behavior. During the execution of the choreographed service, the Monitor is informed about the messages sent or received by the cooperating Web Services and about their execution state. The Monitor utilizes this information to check whether the overall service correctly progresses, i.e., if the message flow among the Web Services is consistent with the choreography. If any fault occurs, the Monitor evaluates whether the choreographed service can still complete and informs the Web Services, in order to let them react to the occurred problem. For generality purposes, we assume that the Monitor does not have any information about the internal implementation of the Web Services. Thus, our framework relies on the analysis of messages.

Our choreography framework builds on WS-Coordination [2] to manage the coordination context between the Web Services but it replaces the Web Services Coordinator with a Web Services Monitor which proactively checks the progress of the choreographed service and propagates the coordination information. In the rest of this paper, Section 2 shows the benefits of monitoring on a sample scenario, Section 3 presents our framework, and Section 4 discusses some related work and concludes the paper.

II. Motivating Example

Monitoring the progress of a choreographed service and notifying the cooperating Web Services about execution problems enables to avoid delays in execution and to react to faults in timely fashion. In fact, the data dependency chains among different Web Services may be long and a Web Service might become aware that it is affected by an execution problem occurred in another Web Service after several messages have been sent. We show this point on a simple example from a Business to Consumer (B2C) e-commerce scenario.

In a typical B2C service, the online store is the contact point with the end customer; given the customer's orders, the store invokes a warehouse to prepare the requested parcel. In turn, the warehouse service might inquire with some shippers about the cost of sending the parcel to the

T. Sobh (ed.), Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, 283–288.

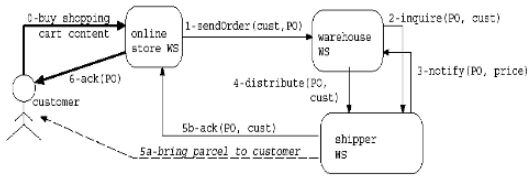

Fig. 1. Collaboration Diagram of a choreographed service.

customer in order to choose the most convenient one. Thus, the online store would not directly control the shipper's activities. However, in order to be able to inform the end customer about possible delivery problems (e.g., delays), the online store needs an acknowledgement from the selected shipper service as soon as the parcels have been sent out. Figure 1 shows the messages exchanged by the cooperating Web Services in the described scenario.

Now, suppose that, after having received an order from the warehouse, the selected shipper service becomes unavailable because of a software problem. Then, the online store would wait for the shipper's acknowledgment until a possibly long time out expires, and it would not be able to inform the end customer about the delay in a timely fashion. By monitoring the situation, the time out might be anticipated and the online store might be informed about the problem as soon as possible.

This example is deliberately simple but it should be noticed that, in a realistic scenario, the number of cooperating Web Services might be large and the dependency graph very complex. Thus, several service consumers might be blocked by the failure of a supplier. Having a global view of the situation is therefore strategic to notify the services which cannot continue their own execution in due time.

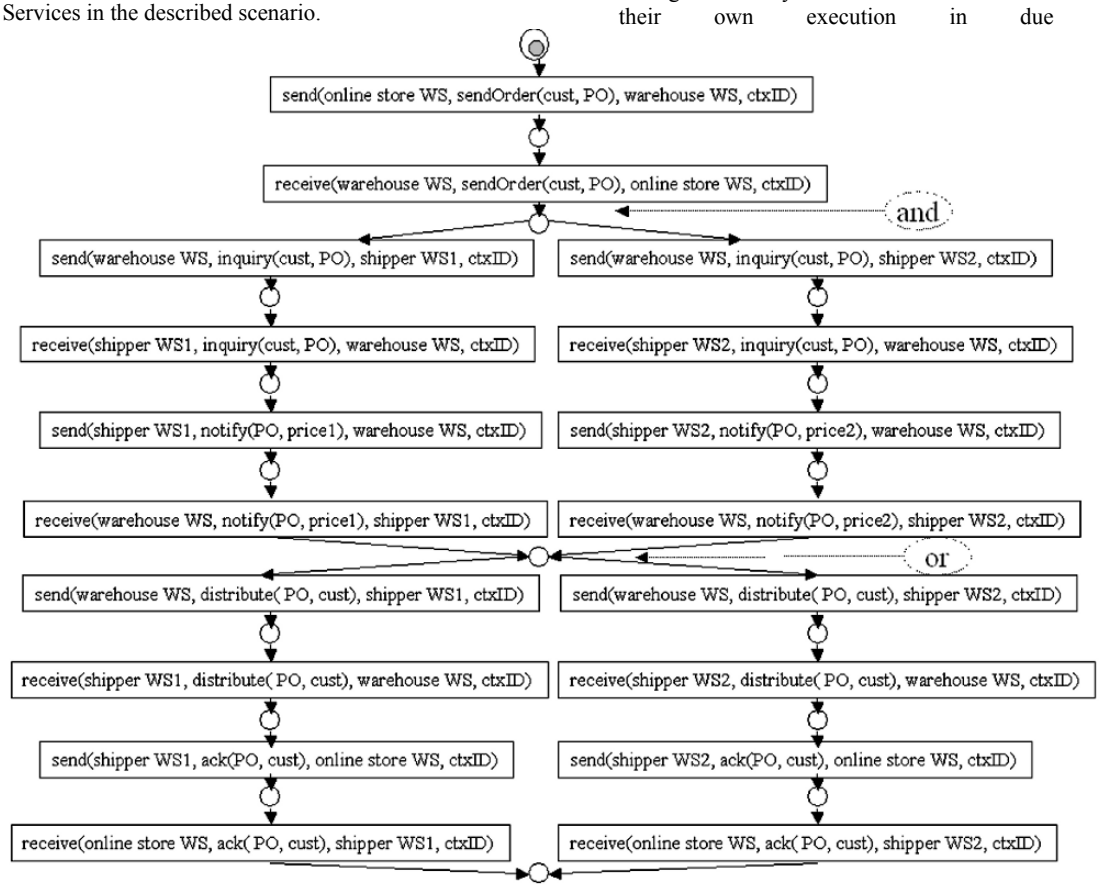

Fig. 2. Choreography specification of the online store example.