ServiceMosaic: Interactive Analysis and Manipulation of Service Conversations

H. Motahari-Nezhad, R. Saint-Paul, B. Benatallah
University of New South Wales, Australia
{hamidm, regiss, boualem}@cse.unsw.edu.au

F. Casati
University of Trento, Italy
casati@dit.unitn.it

J. Ponge & F. Toumani
Blaise Pascal University, France
{ponge,ftoumani}@isima.fr

1. Introduction

In service-oriented computing, a *conversation* is a sequence of message exchanges between two or more services to achieve a certain goal, for example to order and pay for goods. A *business protocol* of a service is a specification of the possible conversations that a service can have with its partners [1]. Motivated by the goal of facilitating the scalable development and maintenance of service oriented applications, especially in light of the many benefits of protocols, we have developed *ServiceMosaic* (servicemosaic.isima.fr), a platform for Web services lifecycle management [2]. *ServiceMosaic* is an interactive and model-driven CASE tool for managing Web service interactions, which consists of two broad modules: *protocol discovery* and *protocol management*.

The discovery module [4] allows deriving protocol models from real-world service conversation logs and refining them. This is important as protocol specifications may not be always available. Furthermore, even when the model is available, discovery allows for checking if the implementation faithfully follows the designed model, and especially, over time, the risk of inconsistencies between specification and implementation increases and so does the importance of deriving the actual protocol model.

The protocol management module [2, 1] is based on an algebra and operators for analysis of protocols. Once users have obtained protocol models, either via discovery or via user-defined specifications, they can use the operators to analyze protocol specifications in different ways, for example to understand which conversations can or cannot occur between two parties, to analyze the protocol differences between two versions of a service, or to understand if a service protocol can support all the conversations as mandated by a standard specification.

This layered and interactive analysis and manipulation of conversations and protocols makes systematic discovery of service protocols and their manipulation a closed loop where users can gradually explore and understand service interactions as well as manipulate service protocols.

2. ServiceMosaic: Design and Implementation

ServiceMosaic framework features a simple, high level but expressive enough model to represent protocol abstractions that are useful and needed in practice [2]. This model

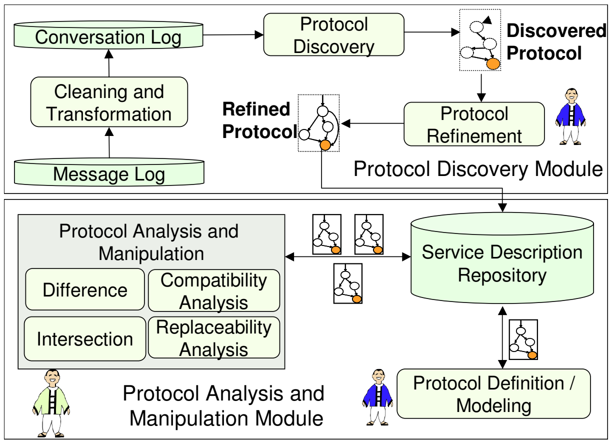
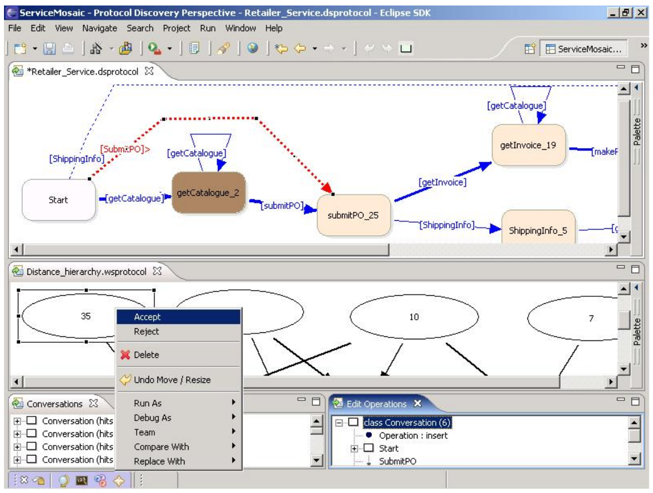

Conversation Log
Protocol Discovery
Discovered Protocol
Cleaning and Transformation
Refined Protocol
Protocol Refinement
Message Log
Protocol Discovery Module
Protocol Analysis and Manipulation
Difference
Compatibility Analysis
Intersection
Replaceability Analysis
Service Description Repository
Protocol Analysis and Manipulation Module
Protocol Definition / Modeling

Figure 1. The architecture of *ServiceMosaic*

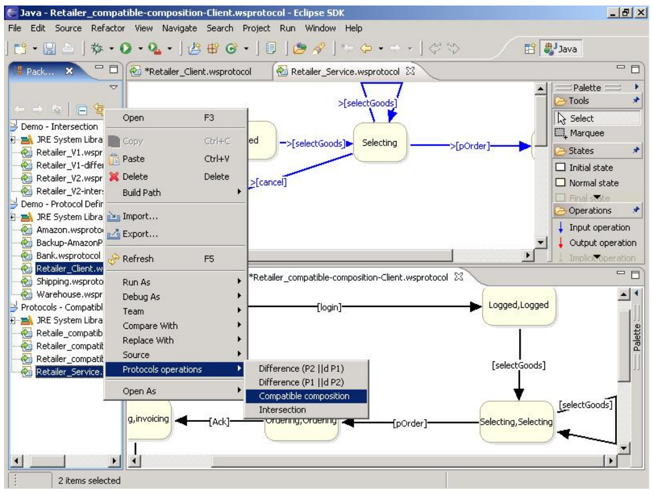
builds upon the traditional state-machine formalism [3] to represent message choreography constraints and extends it to cater for relevant protocol abstractions such as temporal constraints or transactional implications. Figure 1 shows the architecture of *ServiceMosaic* prototype. The core functionalities of *ServiceMosaic* are implemented as Java^{TM} (JDK 1.5) libraries, which are leveraged in GUI tools. All components of *ServiceMosaic* are implemented as Eclipse (www.eclipse.org) plug-ins. These components include the following libraries and GUI tools: library and editor for business protocols enabling protocol definition, library and editor for protocols analysis and manipulation operators (i.e., difference, intersection, matching, compatibility and replace-ability analysis), and a library and editor for protocol discovery and refinement. We have used HP SOA Manager (managementsoftware.hp.com/products/soa/), which is a commercial Web services monitoring tool, to generate message logs used for discovery.

3. Demo Scenarios

Protocol Discovery. We take as a demo scenario one of the examples provided by the Web service Interoperability (WS-I) Organization (www.ws-i.org), related to a retailer supply chain service (called `Retailer` in the following). In the demo we present a `Retailer` service that interacts concurrently with several clients. The retailer company has three objectives: first, it is interested in understanding the actual protocol followed by the implemented `Retailer` ser-



(a) Protocol Discovery and Refinement Editor



(b) Protocol Analysis and Manipulation Editor

Figure 2. Screenshots of *ServiceMosaic*

vice; second, it wants to understand if and how this differs from the one specified by WS-I; third, it wants to analyze if the service can interact with the Web services of prospective partners. To achieve the first goal, we deploy HP SOA Manager to monitor and intercept the message exchanges of `Retailer` with its clients in a simulated environment of real-world. Next, the protocol discovery algorithm is applied [4]. As real-world logs are often imperfect (noisy and incomplete), we show the two distinct stages of the algorithm: (i) the log analysis stage to estimate noise, and (ii) the derivation of the protocol model from an incomplete log.

Protocol Refinement. We show that the user can visually examine and refine the discovered protocol. We illustrate how meta-data associated to protocol model can be used to understand potential errors made during the discovery process and correct them (top of Figure 2(a)). We demonstrate how the user may explore the edit distance hierarchy and examine proposed edit operations to *DP* (middle of Figure 2(a)). The user is recommended to browse nodes of hierarchy based on their cardinality, which gives an indication of the number of conversations that would be accepted by *DP* if the correction is made. We also show that, as an alternative, if users already know the protocol model and are confident that the implemented service matches what they think the protocol model is, then they can directly define the protocol model via the protocol definition editor provided in *ServiceMosaic*.

Protocol Analysis and Manipulation. After discovery and refinement, a protocol modeling the `Retailer` service is obtained. To understand similarities and differences of the discovered protocol with the one published by WS-I, the user next performs replaceability analysis between the two. This analysis is done by applying two operators: intersection and difference. The intersection will tell the user the parts of the two protocols that are similar (support the same conversations), while the difference will stress the conversations supported by one protocol but not by the other (see left-top view of Figure 2(b)). In performing intersection analysis, the user will see a *combined protocol model*, which is again a FSM but where states corresponds to a pair of states, one from each protocol model being analyzed. The combined model defines the conversations accepted by both protocols and also shows in which state each of the two protocols transition to, when a given message is sent or received. Finally, we assume that the `Retailer` wants to interact with a prospective partner, who has made its protocol definition available. To understand which conversations can or cannot occur between the retailer service and the prospective partner, the user applies the compatibility checking operator (bottom of Figure 2(b)). The user will see the result of the operation, again shown as a combined protocol model, showing the possible conversations and how the client and service change state each time a message is sent by one and received by the other.

References

[1] B. Benatallah, F. Casati, and F. Toumani. Representing, analysing and managing web service protocols. *Data and Knowledge Engineering (DKE)*, 2005.

[2] B. Benatallah, F. Casati, F. Toumani, J. Ponge, and H.R.Motahari-Nezhad. ServiceMosaic: a platform for model-driven analysis and management of web services. *IEEE Internet Computing*, 10(4), 2006.

[3] D. Harel and M. Politi. *Modeling Reactive Systems with Statecharts: The Statemate Approach*. McGraw-Hill, Inc., New York, NY, USA, 1998.

[4] H. Motahari, R. Saint-Paul, B. Benatallah, and F. Casati. Protocol discovery from imperfect service interaction logs. In *International Conference on Data Engineering (ICDE'07)*, April 2007.