Using $\mathcal{CEP}$ Technology to Adapt Messages Exchanged by Web Services*

Yehia Taher[1,2], Marie-Christine Fauvet[2], Marlon Dumas[3], Djamal Benslimane[1]
[1] Claude Bernard University of Lyon
LIRIS, 23 bd 11 Novembre, 696222 Villeurbanne cedex, France
[2] University of Grenoble, LIG (MRIM)
385 rue de la bibliothèque – B.P. 53 – 38041 Grenoble Cedex 9, France
[3]Institute of Computer Science, University of Tartu
J Liivi 2, Tartu 50409, Estonia
Yehia.Taher@liris.cnrs.fr

ABSTRACT

Web service may be unable to interact with each other because of incompatibilities between their interfaces. In this paper, we present an event driven approach which aims at adapting messages exchanged during service interactions. The proposed framework relies on Complex Event Processing ($\mathcal{CEP}$) technology, which provides an environment for the development of applications that need to continuously process, analyse and respond to event streams. Our main contribution is a system that enables developers to design and implement $\mathcal{CEP}$-based adapters. These latter are deployed in a $\mathcal{CEP}$ engine which is responsible for continuously receiving messages and processing them according to rules implemented by the adapters. Resulting transformed messages are thus forwarded to their original service recipient.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*

General Terms

Design, Experimentation, Management

Keywords

Web service adaptation, CEP technology

1. INTRODUCTION

The interface of a Web service is defined as the set of messages it can send and receive and the inter-dependencies between them. It is a contract the service and its partners must respect. When an existing service has to interact with another one, their interfaces are more likely to be incompatible [**?**]. When this happens, they are two options: (i) force one of the services to support a new (compatible) interface. This solution is usually unsatisfactory because services then have to expose as many interfaces as interactions they are involved in. (ii) supply an adapter responsible for receiving, adapting, and transmitting messages from one service to the other despite incompatibilities of their interfaces [**?**]. Implementing ad-hoc adapters is a costly and tedious task. Moreover, most of the time an adapter is specific to a pair of services and cannot be reused [**?**].

The study presented in this paper aims at designing a framework for specifying and generating Web service adapters. Our contribution is twofold: (i) The proposed framework supplies operators corresponding to different situations of adaptation, and these operators may be composed to deal with more complex incompatibilities of interfaces. The resulting reusable mapping is then compiled and deployed into a $\mathcal{CEP}$ engine. (ii) We defined a methodology for using $\mathcal{CEP}$ technology in a Service Oriented Architecture.

*This work is funded by the project Web Intelligence granted by the French Rhône-Alpes Region. Most of it was done when Yehia Taher was visiting the BPM Group at QUT, Brisbane.

WWW 2008 April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04

2. EVENT-DRIVEN ADAPTATION

2.1 Illustration

As a motivating example we consider a service S_{sel} intended to offer products for sale, and a service S_{cons} which is meant to place orders against S_{sel}. While S_{cons} was designed in such a way an order of n items is realised by sending n messages, S_{sel} expects an order to be sent at once. To deal with the resulting incompatibility the behaviour of the required adapter is as follow. Messages sent by S_{cons} are received and stored, and messages related to the same order are grouped together. When a condition is satisfied, meaning that an order is completed, an aggregation operation is applied on all related messages. This process results in forwarding to S_{cons} a message that contains the order as expected by S_{sel}'s interface. Since multiple conversation could coexist between S_{cons} and S_{sel}, there is a need to correlate message belonging to the same conversation. Figure **??** depicts such an adaptation process.

2.2 Approach

An Event Driven Architecture ($\mathcal{EDA}$) is a software architecture paradigm promoting the production, detection, consumption of, and reaction to events. Event consumers subscribe to an intermediary event handler, and event producers publish to this handler. When an event is received, the event handler forwards it to the consumer. While $\mathcal{EDA}$ deals with patterns of simple and ordinary events, Complex Event pro-

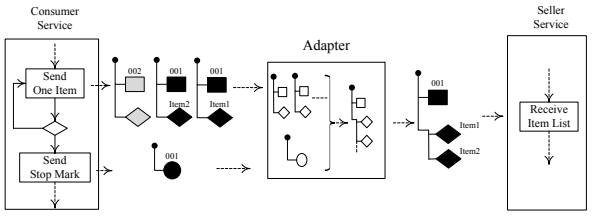

Figure 1: Adaptation Example

cessing[1] ($\mathcal{CEP}$) is a new $\mathcal{EDA}$ paradigm that allows of simple events to be aggregated into complex ones. More precisely, $\mathcal{CEP}$ is a platform for building and running event-based applications that could continuously process event streams to detected a specified confluence of events, and trigger a specific action when the events occur. Most of existing $\mathcal{CEP}$ platforms provide a continuous computation language for specifying queries each of which defining the schema of an event stream to be monitored, defining event patterns to be detected, specifying processing functions and sequencing, and finally declaring output to be generated.

As Web service interactions rely on message exchanges, the process of adaptation consists in consuming incoming messages, processing them, and finally producing messages. With this setting, $\mathcal{CEP}$ fulfils Web service adaptation requirements. Exchange messages are modelled as events, the message adaptation logic is encoded in terms of continuous computation queries ($\mathcal{CCQ}$), that allow the CEP engine to observe message events and detect the predefined adaptation situations. By doing so, a $\mathcal{CEP}$ engine hosting the adaptation $\mathcal{CCQ}$, is able to intermediate messages interactions between Web services, and acts as a Web service adapter.

3. SYSTEM OVERVIEW

As part of our contribution, we provide developers with a framework that allows designing, building and deploying $\mathcal{CEP}$ -based Web service adapters. Figure **??** depicts the proposed framework architecture which is discussed below.

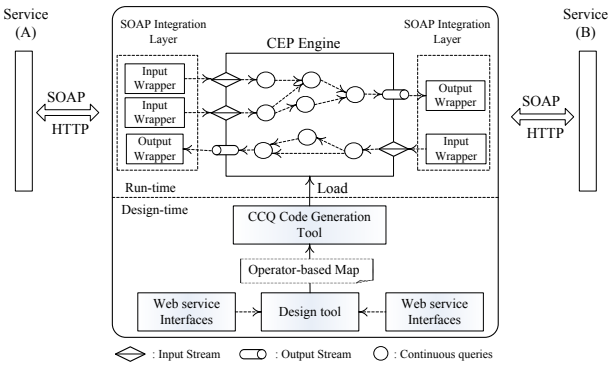

Figure 2: Framework Architecture

3.1 Design-Time environment

We have designed a set of template adaptation operators, where each of them is intended to deal with a specific adaptation situation. More complex adaptation situations could be addressed by composing these operators. Designers of adapters have to identify which adaptations need to be performed between two services, and then wrap the corresponding composition of operators into a `map`. A graphical user interface meant to ease this task is supplied. Subsequently, maps are exported to the $\mathcal{CCQ}$ `code generation tool`, which integrates a `compiler` component used to produce a $\mathcal{CEP}$ `execution-time` module. Eventually, maps are loaded into the $\mathcal{CEP}$ execution engine. We model each operator by a specific automaton therefore easing verification of operator composition correctness. This latter is then compiled and deployed as a $\mathcal{CEP}$ module.

3.2 Run-Time Environment

The run-time environment consists of a $\mathcal{CEP}$ infrastructure including a $\mathcal{CEP}$ `execution engine` and a set of $\mathcal{SOAP}$ Integration layers, which allow a $\mathcal{CEP}$ engine to be integrated in a Service Oriented Architecture. The $\mathcal{CEP}$ engine is embedded as the most important component in the system architecture. It provides the core service of receiving, correlating, analyzing and processing messages against loaded $\mathcal{CCQ}$. Upon a situation detection, events such as message manipulation and production, are generated in response. While Web services communicate through the use of $\mathcal{SOAP}$ messages, the intermediate adapter should be able to consume, analyze, and forward $\mathcal{SOAP}$ messages. To deal with this issue, we have designed a $\mathcal{SOAP}$ Integration Layers model meant to wrap the $\mathcal{CEP}$ infrastructure. Such a model consists of input and output wrappers. A $\mathcal{SOAP}$ input wrapper is used to consume $\mathcal{SOAP}$ messages issues from an output action at the sender service interface. After receiving a message, the input wrapper is meant to transform it to a representation appropriate to the $\mathcal{CEP}$ engine, and then publish it into the engine. Similarly, an output wrapper gets produced event from the engine, transform it in a $\mathcal{SOAP}$ message and then forward it to the corresponding input action at the final receiver service.

4. CONCLUSION

In this paper, we have integrated $\mathcal{CEP}$ technology with Service Oriented Architecture. In particular, we have described how to use an event driven system to adapt message interactions between Web services with incompatible interfaces. An ongoing study, still relying on the $\mathcal{CEP}$ technology, aims at addressing more complex adaptation challenges in service oriented architecture, such as dealing with incompatibilities between Web service interaction protocols.

5. REFERENCES

[1] B. Benatallah and et al. Developing adapters for web services integration. In *Proc. of the 17th CAISE Conf.*, pages 415–429. Springer Verlag, June 2005.

[2] M. Dumas, M. Spork, and K. Wang. Adapt or perish: Algebra and visual notation for service interface adaptation. In *Proc. of the 4th BPM Conf.*, pages 65–80, Vienna, Austria, 09/2006. Springer Verlag.

[3] Y. Taher, D. Benslimane, M.-C. Fauvet, and Z. Maamar. Towards an approach forweb services substitution. *ideas*, 0:166–173, 2006.

[1]http://www.complexevents.com/