Towards Web Services Interaction Mining Architecture for e-commerce applications analysis

ROBERT GOMBOTZ[1], KARIM BAÏNA[2], and SCHAHRAM DUSTDAR[1]

[1] Distributed Systems Group, Vienna University of Technology, AUSTRIA
[2] ENSIAS, Université Mohammed V - Souissi, B.P. 713 Agdal Rabat, MOROCCO
r.gombotz@infosys.tuwien.ac.at, baina@ensias.ma, dustdar@infosys.tuwien.ac.at

Abstract: - Nowadays, Web service technology plays an increasing role in internet applications, in general, and e-commerce applications, in particular. In fact, service-oriented systems can be expected to grow larger in complexity. Such large systems demand for tools that allow for analyzing and monitoring of service-oriented systems in use. Our work attempts to draw the necessary architecture in order to analyze interactions between Web service consumer and provider. WSIM modelling architecture is built over three layers : Web service operations, interactions and workflows. The paper aim is to present WSIM modelling architecture and how it could be implemented to support existing Web service applications (e.g. e-commerce applications).

Key-words: e-Commerce Services, Web service interactions, Web service logging, Web service mining

1 Introduction

It becomes obvious that Web service (a.k.a. WS) technology will be indispensable in building and integrating internet applications (e.g. CRM, SCM, e-commerce applications) [1]. Those service-based applications are expected to grow larger in complexity. In order to make WS applications easier to use and maintain for providers and customers, interesting challenges are highlighted:

- *Discovering complex patterns within Web Services applications* (e.g. identifying reconfigurable web service architectural patterns, runtime web service behavioural patterns)
- *Supervising and monitoring of Web Services applications* (e.g. by building analysis and administration scoreboards for web-service based applications)

Within this problematic, our work propose modelling and implementation architectures for Web Service interactions analysis. Mining describes the process of discovering knowledge in large amounts of data (in our case Web Service applications data). Our contribution is not specific to a mining technique; it proposes generic architectural bricks so using of specific mining technique could be always possible.

The remainder of this paper is structured as follows. Firstly, we present our WSIM modelling and implementation architecture, and finally we discuss related works before concluding.

2 Web Services Interaction Mining – Modelling Architecture

We develop our Web Services Interaction Mining (a.k.a. WSIM) approach with regards to three levels of abstraction that represent three complementary Web services ”views”. Figure 2 depicts a stack of views on Web services. As one goes from the top to the bottom, the level of abstraction falls and we are looking at things in higher detail.

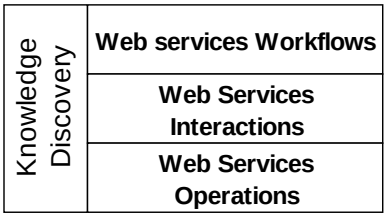

Fig. 2. Web Service Mining Levels

In the following subsections we will detail each of the three WSIM architecture levels.

For each level, we present a normative format of log entries, or a log specification, as well as simple examples of log records. By normative we mean log formats the way we would like them to be. The focus is not yet on how to keep these logs or on who should provide them. This issue will be discussed in the next section thereafter.

2.1 Web service Operations Mining level

On the Web service Operation level, we want to examine only one single Web service and its internal behaviour (e.g. interface, conversation protocol [2]). We will not concern ourselves with a Web service's interactions with other Web services or applications, but rather focus on its functionality as if it were alone in the world. Furthermore, the focus might even be on just one operation of the Web service. However, we also want to examine the Web service as a whole. Relating this to mining, a given log output of the Web service shall be analyzed to gain information about its behaviour. Figure 4 shows Web Service Operation Log Model. Each OperationEvent of this OperationLog is described by an Activity (i.e. operation name), a Performer (the Web Service client), a status (either Start or Complete), and a TimeStamp (the current datetime).

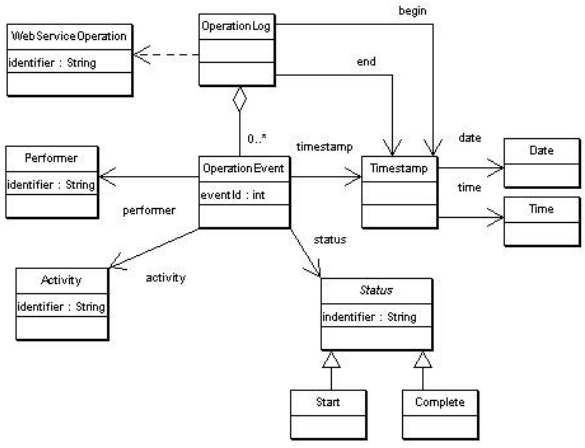

Fig. 4. Class diagram of operation log Model

The following example gives a Web Service OperationLog sample according to Web Service Operation Log Model shown in Figure 4 :

Start - acceptOffer - customer05 - **2003:01:02**:15:45
Complete - acceptOffer - customer05 - 2003:01:02:15:46
Start - executeCooperation - customer05 - 2003:01:28:06:00

Complete - evaluateOffer - customer09 - **2003:01:07**:10:23
Start - rejectOffer - customer09 - 2003:01:07:14:03
Complete - executeCooperation - customer05 - 2003:07:27:18:00

2.2 Web Service Interactions Mining level

On the Web service Interaction level, we again focus our attention to one Web service, but also want to take into account its "direct neighbours". The term "direct neighbours" refers to other Web services that the examined WS interacts with. Such interactions may be explicit, i.e. defined in a composition language (e.g. BPEL), or implicit, i.e. calls to other WS from within the Web service's implementation. Explicit interactions are also said to be *declarative*, implicit interactions are also called *programmatic*. On this level we want to mine log data for further information about the Web service's interactions with others. This information could reveal interesting facts about a Web service's interaction partners, such as critical dependencies. In order to be able to mine for programmatic interactions between WS we need those interactions to be logged. The four basic types of interactions between Web services are *one-way*, *request-response*, *solicit-response* and *notification*. Figure 7 shows Web Service Interaction Log Model. Each InteractionEvent of this InteractionLog is described by an identifier marking it as an interaction entry, an identifier of whether the interaction is one-way (asynchronous) or two-way (synchronous), an identifier of where within the interaction we are, an identifier concerning the interaction partner, an identifier of the activity that is being performed, and a timestamp. The identifier marking the type of log entry could be simply "int", which is short for interaction. The following identifier can be "sync", stating that the interaction is two-way, or

synchronous. The next identifier should mark the "state" of the interaction and can be one of the following : "sendRequest" indicates that the interaction was just initiated; "receiveRequest" (logged by the second entity involved) states that the request was received. That second entity would then declare it has replied by logging "sendResponse". Finally, the initiator of the interaction would log "receiveResponse".

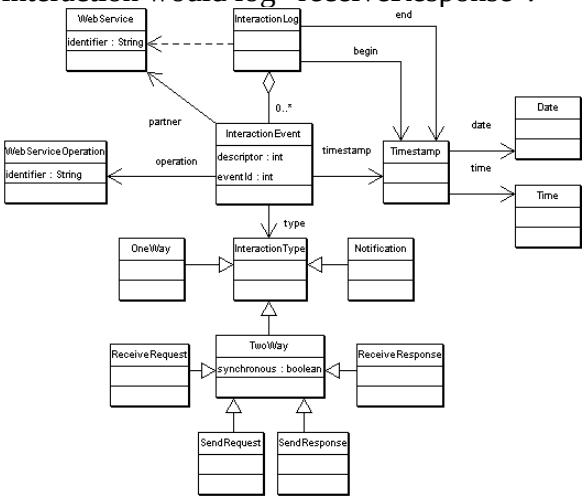

Fig. 7. Class diagram of interaction log Model

Figure 8 depicts an interaction graph for Web service A, i.e., the result of mining on the Web services Interactions level with the focus on Web service A. It contains one example of each of the four basic types of interactions between WS. One-way interaction is depicted as a unidirectional, lined arrow pointing from the initiator to the called WS. Two-way interaction is depicted as a bidirectional lined arrow.

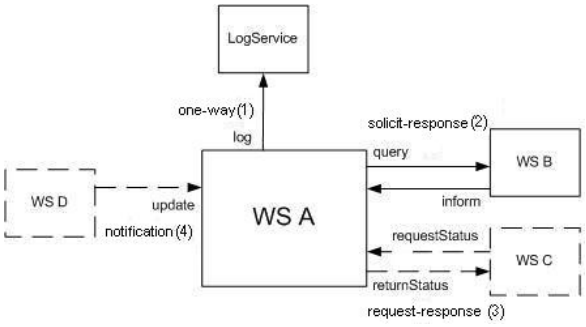

Fig. 8. interaction graph for Web service A

Interaction (1) between A and LogService is one-way. Interaction (2) between A and B, is of type solicit-response. Interaction (3) between A and C is of type request-response, and is initiated by C. Interaction (4) between A and D in a notification, where D sends an unanswered message to WS A. In the following we give the four Web Service InteractionLogs according to Web Service Interaction Log Model shown in Figure 7.

A InteractionLog
async - oneWay - log - LogService - 2003:01:02:7:03

sync - sendRequest - query - B -2003:01:02: 8:01
sync - receiveResponse - query - B - 2003:01:02:8:04

async - receiveRequest - returnStatus - C - 2003:01:02:9:02
async - sendResponse - returnStatus - C - 2003:01:02:9:03

async - notification - update - D - 2003:01:02:10:02

LogService InteractionLog
async - notification - loggingRecord - A - 2003:01:02:7:04

B InteractionLog
sync - receiveRequest - inform - A - 2003:01:02:8:02
sync - sendResponse - inform - A - 2003:01:02:8:03

C Interaction Log
async - sendRequest - requestStatus - A - 2003:01:02:9:01
async - receiveResponse - requestStatus - A - 2003:01:02:9:04

D Interaction Log
async - oneWay - sendUpdate - A - 2003:01:02:10:01

2.3 Web Service Workflows Mining level

The highest level of abstraction is the Web service Workflow level. As the name suggests, the focus is on large-scale interactions and collaborations of Web services which together form an entire workflow. The interaction graphs built through WSIM on the Interaction level do not contain any information on workflows. They display all interactions of a WS, no matter what workflow they belong to. On this level, we want to examine the execution of the entire process. Here we will be able to benefit from the results and findings of researchers in the field of process mining. Even though our current focus is on mining, it must be stressed that once a mining effort is completed (with respect to its primary goals of building a model of a process), it

should be continued and serve the purpose of monitoring. Therefore, future log data should constantly be analyzed and compared to the model established in the initial mining process. One might find exceptions in future behaviour of the examined system, or - in an even more undesirable case - find that the initial model was built on false assumptions, possibly because of insufficient log data.

On the workflow level, we want to apply process mining to a service-oriented system. To do that, our log specifications need to be extended. In order for workflow mining to be possible, log entries need to include workflow information. Figure 4 shows Web Service Workflow Log Model. Each WorkflowEvent of this WorkflowLog extends a InteractionEvent by referring a processID, and an instanceID. The processID specifies the workflow, or business process that is currently being executed.

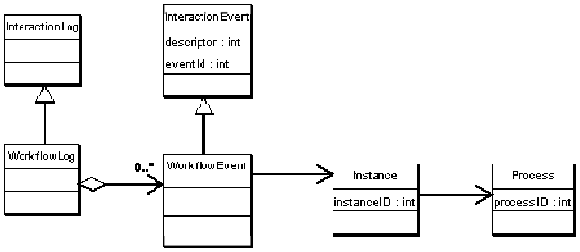

Fig 9. Class diagram of workflow log Model

In the following we give A Web Service WorkflowLog according to Web Service Workflow Log Model shown in Figure 9.

A WorkflowLog (for processID = 101 and instanceID = 13)

101 - 13 - async - oneWay - log - LogService - 2003:01:02:7:03

101 - 13 - sync - sendRequest - query - B -2003:01:02: 8:01
101 - 13 - sync - receiveResponse - query - B - 2003:01:02:8:04

101-13-async-receiveRequest-returnStatus - C - 2003:01:02:9:02
101-13-async-sendResponse-returnStatus - C - 2003:01:02:9:03

101 - 13 - async - notification - update - D - 2003:01:02:10:02

Reconsider now our four Web Srevices A-D, each performing a single, independent task. The sum of these tasks might be a business process. Every entity involved in the process provides event-based data, which is logged. An event occurs, when an activity is started and when it is completed. In a simple case, the data provided should consist of the identifier of the activity that is being performed, the event type, which can be ”started” or ”completed”, and a time stamp. After a sufficient amount of log data has been collected one can mine this logging information for patterns and thereby find that e.g. an activity A is always performed before activity B. Activity B in turn is always performed before activity C, and sometimes, but not always before D. Activities B and D are also always completed before the execution of C starts. From this information one could derive the simple workflow model shown in Figure 10.

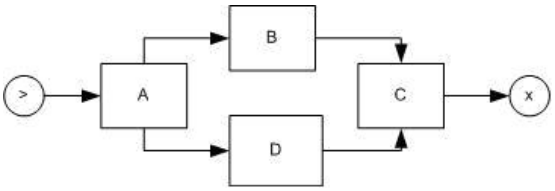

Fig. 10. Simple workflow model

3 Web Services Interaction Mining – Implementation Architecture

In this section we focus on discussing implementation aspects. Figure 12 details WSIM components and their coordination as follows:

1. Event Sensors: Starting from Web services executions and interactions, events are intercepted by different log sensors and gathered into logs. Figure 11 gives a more detailed look on an interaction layers between two WS. It also contains the involved entities, i.e., the Web services, the WS containers that manage the WS, the hosts the WS containers are running on, and the SOAP message that is being exchanged between the Web services. One of our core assumptions is therefore that the WS we

want to mine use SOAP (over HTTP) to communicate, or interact. Within this context, event sensors may be from three different levels:

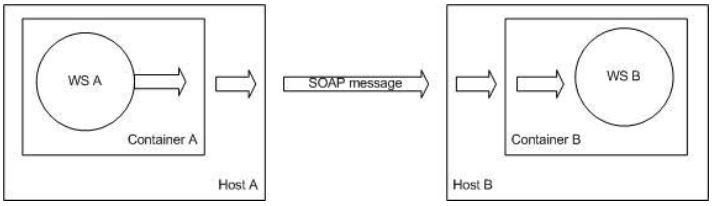

Fig. 11. Web Service Interaction layers between two Web Servces

(a) *Web server (host) level:* Web service interaction messages are intercepted at HTTP-listener level. This approach allows to monitor all incoming requests to the Web server engine and their corresponding responses (e.g. with tools like TCP Tunnel/Monitor tool as part of the Apache SOAP package3[1]). An advantage of this approach is that it is 100% platform/vendor independent. WSs (using SOAP over HTTP) deployed in any engine could be monitored. The disadvantage is that, at this general level, other web resources might be deployed on the Web server engine, and we need to select only SOAP messages and neglect other HTTP messages that we do not care about;

(b) *Web service container level:* Web service interaction messages are intercepted at the SOAP-listener level before they are passed to the web service (e.g. with tools like Mindreef's SOAPscope 3.04[2]. An advantage of this approach is that all (and only) SOAP messages are intercepted. A clear disadvantage is that this approach is 100% platform/vendor dependent. Actually, the event sensor will depend on the SOAP-listener and its hosting environment with the Application server (e.g. the specific servlet SOAP-listener and the Java Virtual Machine within Tomcat5[3]);

(c) *Web service level:* Web service interaction messages are logged by the Web service itself. Actually, the WS passes the SOAP message to the event sensor directely through its API. An advantage of this approach is that the WS logs additional information that will make Web service Mining easy (e.g. adding the Web service related workflow "processID" and "instanceID" necessary in Workflow mining level within the logged SOAP-message header). Another advantage is that, since Web service logs itself SOAP-messages in its known event sensor, the Web service will provide plain-text instead of eventual encrypted SOAP-messages that will be hardly used by the Web service Logger and Miner. A clear disadvantage is that this approach is 100% Web service dependent, and needs additional implementation efforts. Moreover, interactions of WSs developed prior to WSIM could not be intercepted by this kind of event sensors.

[1] The Apache Software Foundation, ws.apache.org/soap
[2] Mindreef SOAPscope, www.mindreef.com/products/overview.html
[3] The Apache Software Foundation, jakarta.apache.org/tomcat

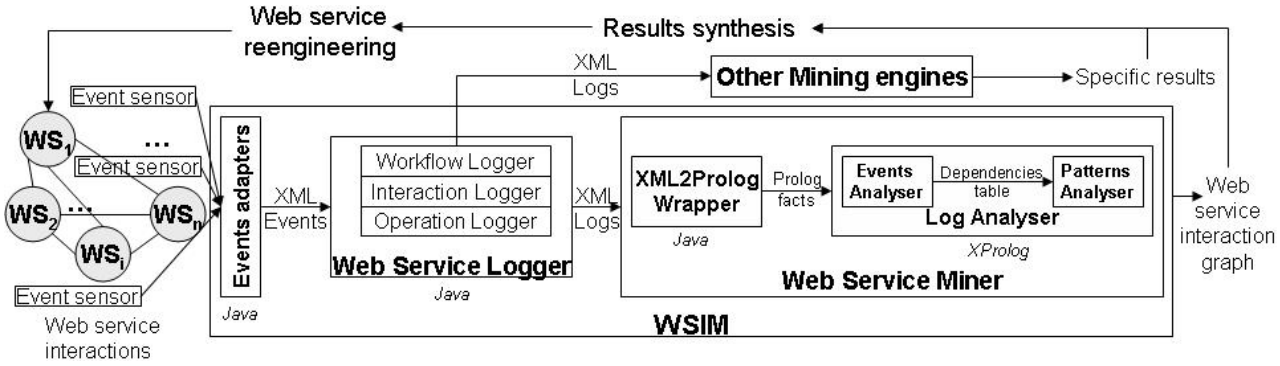

Fig. 12. WSIM Components Overview

2. Event Adapters: To keep the intercepted events homogeneous, and usable by to our web Service log model, and adaption step is necessary. Actually, event adapters translate those non-structured web service events into our web Service log model compliant XML structures;

3. Web Service Logger: In this step, XML formatted events are filtered according to their abstraction level into three different logs implementing operations, interactions, and workflow log models;

4. Web Service Miner: Till this step, we possess XML structured logs that can be analysed by any mining tool and particularly our Web Service Miner that is an extension of WorkflowMiner [8, 3, 4]. Our Web Service Miner has two components XML2PrologWrapper and a LogAnalyser that specify WorkflowMiner context with Web service interaction graph context:

- **(a)** *XML2Prolog Wrapper:* To enable events log to be easily minable in our Web Service Miner, these logs are wrapped into a common 1st order logic format, compliant with UML class diagrams shown in figures 4 and 7;
- **(b)** *Log Analyser:* Mining rules are applied on resulted 1st order log events to discover Web service interaction patterns that are synthesised into a Web service interaction graph. We use a Prolog-based presentation for log events, and mining rules using the XProlog system6[4];

5. Since a Web service interaction graph is discovered, the Web service designer will have a look on the developed web service to restructure or redesign his Web service interactions.

4 Related works

Valuable research results have been achieved in data mining, process mining, and web mining. However, the idea of Web service Interaction mining as proposed in this paper, is yet a new hot research topic. In this section, we discuss process mining works that are the most relevant to our area. Process mining is the major issue in WSIM on the Workflow level. [5] provides an overview of the ideas behind process mining, or workflow mining. They describe process mining as ”a method of distilling a structured process description from a set of real executions”.

Also, the major challenges in process mining are discussed in detail, which gives the reader a very good idea of the problems one might be faced with. These challenges are e.g., mining hidden tasks, mining non-free-choice constructs or loops, dealing with noise and incompleteness or gathering data from heterogeneous sources. Furthermore, an overview over different mining algorithms is given as well as a brief

[4] XProlog, www.iro.umontreal.ca/~vaucher/XProlog/

description of the other papers which are part of this special issue on process mining. In [6], a detailed description of, what the input data should look like in order to allow for the mining of exact workflow models is presented. Some of these specifications are used in section 3 of this paper where we present our suggestions of log specifications. Furthermore, [6] elaborates in detail on a step-by-step description of the workflow mining process itself. This process includes the pre-processing of workflow logs and the building of sub-models.

The theoretical description of the process is followed by an example, which improves the reader's understanding of process mining significantly. Also, an implementation of the algorithm is presented in the form of an application named Process Miner. In order to monitor business process quality, [8] proposes a solution, based on data warehousing and mining techniques for analyzing, predicting, and preventing the occurrence of exceptions. Other works in process mining focus on discovering workflow transactional behaviour among workflow instance through execution log [7].

5 Conclusion and Perspectives

In this paper we have outlined our novel idea of Web Service Interaction Mining (WSIM). We have identified three levels of abstraction with respect to WSIM: the operation level, the interaction level and the workflow level. The term mining implies that available log data should be analyzed to acquire additional knowledge about a system. We believe that developing Web services with consideration for WSIM can significantly improve the manageability of a WS or of an entire service-oriented system. We especially discussed WSIM on the operations and interactions level. The information regarding all interaction partners can be vital during e.g., an impact analysis of changes made to a Web service. In the near future we will therefore direct our attention to developing an easy-to-use framework that allows for the implementation of WS which are ready for WSIM. However, we also want to take into consideration Web services that have already been deployed. In our future work, we will examine standard logging and mining tools and test their integration usability within WSIM. Especially on the Web services interactions level we see some opportunities of mining for Web service interactions, which were discussed in this paper. WSIM on the workflow level seems to pose the greatest difficulties. As we have shown, WSIM on the workflow level does require additional development effort. A workflowID and an instanceID are needed and can only be available if provided by the WS itself.

References

[**1**] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. Web Services Concepts, Architectures and Applications. Data-Centric Systems and Applications, 2004. Springer Verlag.

[**2**] K. Baïna, B. Benatallah, F. Casati, and F. Toumani. Model-Driven Web Service Development. The 16th International Conference on Advanced Information Systems Engineering (CAiSE'04), pp. 290-306, LNCS 3084, Riga, Latvia, June 7-11, 2004, Springer Verlag.

[**3**] W. Gaaloul, S. Alaoui, H. Bakkali, K. Baïna, and C. Godart. WorkflowMiner : An infrastructure for Mining Workflow Patterns. 3mes Journes Nationales sur les Systèmes Intelligents : Théories et Applications (SITA'04), Rabat, Morocco, 6-7 Dec. 2004.

[**4**] W. Gaaloul, S. Alaoui, K. Baïna and C. Godart. Mining Workflow Patterns through Event-data Analysis. The IEEE/IPSJ International Symposium on Applications and the Internet (SAINT'05). Workshop 6 Teamware: supporting scalable virtual teams in multi-organizational settings, Trento, Italy, 31 Jan. - 4 Feb. 2004, IEEE Computer Society Press.

[**5**] W. M. P. Van der Aalst, and A. J. M. M. Weijters. Process mining: a research agenda. Computers in Industry 53, 2003, Elsevier B.V.

[**6**]. G. Schimm. Mining exact models of concurrent workflows. Computers in Industry 53, 2003, Elsevier B.V.

[**7**] D. Grigori, F. Casati, U. Dayal, M-C Shan. Improving Business Process Quality through Exception Understanding, Prediction, and Prevention. (VLDB'01), Roma, Italy.

[**8**] W. Gaaloul, S. Bhiri, and C. Godart. Discovering Workflow Transactional Behavior From Event-Based Log. (CoopIS'04), Agia Napa, Cyprus, 25-29 Oct. 2004, Springer Verlag