Research Challenges and Opportunities In Web Services Mining

Walid Gaaloul\(^{(1)}\), Sami Bhiri\(^{(2)}\), and Claude Godart\(^{(1)}\)

\(^{(1)}\)LORIA-INRIA
BP 239, F-54506 Vandoeuvre-lès-Nancy Cedex, France
\{gaaloul, godart\}@loria.fr

\(^{(2)}\)Digital Enterprise Research Institute
IDA Business Park, Galway, Ireland
sami.bhiri@deri.org

ABSTRACT. Web services are becoming more and more complex, involving numerous interacting business objects within considerable processes. In order to fully explore Web service business opportunities while ensuring a correct and reliable modelling and execution, analysing and tracking Web services interactions will enable them to be well understood and controlled. The work described in this paper is a contribution to these issues for Web services based process applications.

This article describes a novel way of combining data mining techniques on Web Services log in order to discover actionable Web service modelling intelligence. Our work attempts to apply Web service log-based analysis and process mining techniques in order to provide a mean to ensure a correct and reliable Web services modelling and execution. Concretely, we deal, in this paper, with Web services log collecting issue. Then, we present a quick survey on opportunities and challenges to exploit existing process mining techniques on Web services.

KEYWORDS: Web services architecture, Web Services logging, process mining.
1. Introduction

The Web services paradigm promises to enable rich, flexible, and dynamic inter-operation of highly distributed, heterogeneous network enabled services [1]. Currently much research efforts are going into the development of Web service technologies ensuring such requirements. Such quickly evolving systems demand for sophisticated design in order to answer to users' needs and requirements and afterwards guarantee reliable executions. In order to fully explore Web Service business opportunities while ensuring a correct and reliable execution and optimizing correct system behaviour, we find that it is important to track Web services utilization.

In this paper, we will try to introduce the idea of Web services mining that makes use of the findings in the fields of data mining and process mining and apply them to the world of Web services and service-oriented architectures. The transition from vast amounts of Internet server and Web Services transactions data to actionable modelling intelligence would be one of the major challenges in Web Services research field. Basically, we present a different approach that starts from a Web Service executions log and uses a set of mining techniques to discover the Web services model in order to detect and prevent anomalies. Consequently we present a set Web services logging facilities that captures comprehensive services usage information in order to propose a set of tools using these logs to mine Web Services transactions and interactions and providing feedback on Web Service behaviour.

2. Prerequisites

In this section, we present the relevant Web services concepts required to understand the domain in which we operate. Then, we present the purpose and an overview of our approach. Finally we show the different mining levels we distinguish.

2.1. Web services concepts

In this section, we present some relevant web services concepts to settle the different mining levels. The concepts we are considering are: Web service WSDL interface, Web service abstract process interface, Web services choreography and Web service orchestration model.

The precise definition of a “Web service” is in progress. The concept of a “Web service” refers to an application (a program) provision on Internet by a supplier, and accessible by the customers through protocols Internet standards. The consortium W3C defines a Web service as being an application or a software component (i) identified by a URI, (ii) whose its interfaces and its binding can be described in XML, (iii) its definition can be discovered by other Web services and (iv) it can interact directly with other Web services through language XML and by using protocols Internet.

Three major initiatives of standardization were proposed by the W3C
Web Services Mining 3

consortium to support the interactions between the Web services: SOAP, UDDI and WSDL. SOAP is a protocol of transport allowing the exchange of XML documents in a distributed and decentralized environment. UDDI is a specification which defines the mechanisms which make it possible the companies to publish their services and to discover and to interact with other services via Web.

WSDL is a language for Web services capabilities description based on XML. A WSDL document describes what a Web service offers, where it resides and how one can call upon it. WSDL defines a Web service as a set of operations. For example, we consider here an application for an Online Travel Arrangement (OTA for short). This application is externalised as a Web service. WSDL defines the OTA service as the set of the following operations (see Figure 1): Request Customer Specification, Notify Hotel Unavailability, Request/Receive Payment Information, Notify Payment Rejection, and Confirm Travel Arrangement.

WSDL defines a Web service as a set of independent operations. However in a B2B context, a Web service encapsulates an internal application of a company. The operations offered by a service represent then the entry points to this application via which it interacts with the external world. And thereafter, the interaction with a Web service implies an exchange of messages in a well defined order. A Web service abstract process defines the execution logic of the operations defined independently.
Figure 1 illustrates the abstract process of the OTA service. This interface stipulates, among others, that the Request/Receive payment information operation must be preceded by the Receive customer requirements service.

A conversation between a Web service and a client (that may be another Web service) indicates the sequence of messages exchanged between them. A conversation between two Web services occurs with respect to their abstract processes. Figure 1 shows a conversation with the OTA service. Choreography is the coordination of several conversations between various Web services. It is important to note that in choreography a set of Web services collaborate to achieve a common goal without creating a new service. Choreography is managed in a decentralized way by the different partners.

One of the important concepts that is offered by the Web services approach is
the ability to create a new service by combining existing Web services. Web services composition deals with the implementation of a Web service whose application logic involves the invocation of operations offered by other Web services. An orchestration model specifies the executable process of a composite service. For instance, the OTA service can be defined as a composite service. Figure 2 describes its orchestration model.

2.2. Overview of our approach

The main purpose of this work is to apply some mining and conformance techniques as a feedback loop to improve the Web services concepts described above, in particular, the abstract process and the orchestration model.

Generally, formal previous approaches develop, using their Web service modelling formalisms, a set of techniques to analyze the composition model and check some properties (see Figure 3.a). [1] proposes a formal framework for modelling, specifying and analyzing the global behavior of Web services compositions. This approach models Web services by mealy machines (finite state machines with input an output). Based on this formal framework, authors illustrate the unexpected nature of the interplay between local and global composite Web services. In [2], authors propose Petri net-based algebra for composing Web services. This formal model allows the verification of properties and the detection of inconsistencies both between and within services.


Although powerful, the above formal approaches may fail, in some cases, to ensure composite web services reliable executions or the conformance between the orchestration model and the abstract process, even if they formally validate such issues. This is because properties specified in the studied composition models remain assumptions that may not coincide with the reality (i.e. effective composite web services executions).

Back to the OTA service, we suppose that each component service that does not have a recovery mechanism is supposed to be sure to complete. Let us suppose that in reality (by observing several executions) the Document Delivery by Fedex service is not sure to complete (contrary to what is specified in the orchestration model) and the Online Payment service is a pivot service. That means the Document Delivery by Fedex service can fail and the Online Payment service cannot be compensated or even re-activated. Aware about such dysfunction, designers can act to remedy to these anomalies and improve the orchestration model.
In this paper, we propose a log based approach (see Figure 3.b) that starts from the execution log to mine and analyse Web services behaviour. This mining step will be used in future to improve the Web services modelling.

2.3. Mining levels

In contrast with workflow mining research where we find a unique control flow mining level, the Web service mining issue contains several levels due to the complexity and the richness of the composition Web service model. According to the different concepts we introduced above, we judged judicious to distinguish the different levels:

- Web service WSDL interface level 1: In this level, we are interested, in the set of operations offered by the Web service.
- Web service abstract process level 2: In this level, we are interested, in addition to the set of offered operations, in the execution order logic between them.
- Web services choreography level 3: In this level, we are interested, in the set of interactions exchanged within the context of a given choreography.
- Orchestration model of a composite Web service level 4: In this level, we are interested, in the executable process implementing a composite Web service.

In the next section, we detail existing Web services logging facilities. We show
Web Services Mining

in particular their adequacy for some mining levels and their limits for others one. We propose how we can extend them to overcome these limits.

2. Web Service logging and Web service mining levels

In this section we examine and formalize the logging possibilities in service oriented architectures which is a critical need within our approach. Thus, the first step in the Web Service mining process consists of gathering the relevant Web data, which will be analyzed to provide useful information about the Web Service behaviour. We discuss how these log records could be obtained by using existing tools or specifying additional solutions. Indeed, the levels of logging vary in the richness of the information that is logged and in the additional development effort that is needed when implementing the additional features.

Then, we show that the mining abilities, (i.e. the capability to mine one of the presented Web service level) is tightly related to what of information provided in web service log and depend strongly on its richness. Basically, we identified two Web services logging levels: trivial and advanced levels. Trivial level provides a set of existing solutions to capture web services logs. These solutions remain quite poor to mine advanced web service behaviour. That is why, in advanced level, we propose a set of developed techniques that allows us to record the needed information to mine more advanced behaviour. This additional information is needed in order to be able to distinguish between web services composition or choreography instances.

2.3. Trivial level

Web service log collecting solution: There are two main sources of data for Web log collecting, corresponding to the interacting two software systems: data on the Web server side and data on the client side. The existing techniques are commonly achieved by enabling the respective Web servers logging facilities. There already exist many investigations and proposals on Web server log and associated analysis techniques. Actually, Web Usage Mining WUM [9] research works describe the most known mean of web log collecting. Basically, server logs are either stored in the Common Log Format [10] or the more recent Combined Log Format [11]. They consist primarily of various types of logs generated by the Web server. Most of the Web servers support as a default option the Common Log Format, which is a fairly basic form of Web server logging. It tracks different elements of the Web transaction.

Concretely, each request recorded by reporting a line of text, with the different elements of the request separated by spaces (items in quotes or square brackets are considered one item), and items that aren’t sent are listed as a hyphen or dash (-). This log, originally conceived for administrative purposes, stores data, which consists of sequential strings of data containing: the requestors IP address, optional user identification, timestamp, request method (This is the actual HTTP request line as it was sent from the client computer), request status code (The status refers to the
HTTP status codes indicating if the request was successful and so on), sent data (size and number of bytes that were sent in the server's response), authenticated user name, the User-Agent (the name of the user on the client computer, i.e., Axis/1.1). For instance, the log entry recorded in Apache Tomcat when a request is sent to a DocumentDeliverybyDHL Web Service in our motivating example may look as follows:

```
127.0.0.1- -[15/Mar/2005:19:50:13+0100] "POST /axis/services/Document Delivery by DHL HTTP/1.0" 200 819 "-" "Axis/1.1"
```

Related Web service mining level: Traditionally this source of information has been used to support smooth site maintenance or operation, by monitoring Web traffic, checking possible bottlenecks, availability, and customer demand and response times. This log seems to be enough to retrieve information concerning Web service utilization metrics (web service consumers, web service calls, web services failures).

Thus, the related mining level deal with a single Web services internal behaviour and its operations. The mining results are interested in tracking of the service consumer, determining which service is called how often, or analyzing service failure rates that could be categorized in the predefined level 1 (i.e. the Web service WSDL interface). This level does not consider interactions with other Web Services. However to discover other information related to WSDL (web service operations, web input/output messages) methods exchanged capturing SOAP messages is mandatory. The using HTTP listeners (for instance, Apache TCP Tunnel/Monitor [12]) in web server side allow capturing related SOAP messages.

But, the arising paradigm of Web services requires richer information in order to fully capture business interactions and customer electronic behaviour in this new Web environment. The proposed solutions remain quite “poor” to mine advanced web service behaviour. Since the Web server log derives from pages requests accessed by a user, it is not tailored to capture service composition or orchestration. That is why, we propose in the following a set of developed techniques that allows to record the additional information to mine more advanced behaviour.

2.3. Trivial level

Web service log collecting solutions: Successful mining for advanced architectures in Web Services models requires choreography or orchestration information in log records. Such information is not available in conventional Web server logs. Therefore, the highest level of logging must provide for both a choreography or orchestration identifier and a case identifier in each interaction that is logged. This in turn requires for additional design and implementation considerations when implementing service-oriented architectures. To be precise,
Web services must "cooperate" in a sense that they are able to receive and forward information regarding the choreography they are currently part of. The most difficult part of using this method is convincing the users to cooperate in that way.

There are at least two ways to implement a Web services logging facility. One of them, a known method for debugging, is to insert logging statements into the source code of each service in order to call another service or component, responsible for logging. However, this solution has a main disadvantage: we do not have ownership over third parties code and we cannot guarantee they are willing to change it on someone else behalf. Furthermore, modifying existing applications may be time consuming and error prone.

Since all interactions between Web Services happen through the exchange of SOAP message (over HTTP), the other alternative is to use SOAP headers that provides additional information on the message's content concerning choreography. Basically, we modify SOAP headers to include and gather the additional needed information capturing choreography details. Those data are stored in the special <WSHeaders>. This tag encapsulates headers attributes like: choreographyprotocol, choreographyname, choreographycase and any other tag inserted by the service to record optional information; for example, the <soapenv:choreographyprotocol> tag, may be used to register that the service was called by WS–CDL choreography protocol. The SOAP message header may look like:

```xml
<soapenv:Header>
  <soapenv:choreographyprotocol
      soapenv:mustUnderstand = "0"
      xsi:type = "xsd:string" > WS–CDL
  </soapenv:choreographyprotocol>
  <soapenv:choreographyname
      soapenv:mustUnderstand = "0"
      xsi:type = "xsd:string" > OTA
  </soapenv:choreographyname>
  <soapenv:choreographycase
      soapenv:mustUnderstand = "0"
      xsi:type = "xsd:int" > 123
  </soapenv:choreographycase>
</soapenv:Header>
```

Then, we use SOAP intermediaries [13] which are an application located between a client and a service provider. These intermediaries are capable of both receiving and forwarding SOAP messages. They are located on web services provider and they intercept SOAP request messages from either a Web service sender or captures SOAP response messages from either a Web service provider. After Web services should process these information and forward it to other clients Web services. In Web service Client-side, this remote agent can be implemented to
Intermediaries’ interception and SOAP headers modifying technique offers a non-intrusive way to extend the functionality of the client and the service provider log collecting abilities. They also offer flexibility, since they can be dynamically added and removed, without changing the code of existing services. But, the implementation of client-side data collection methods requires user cooperation, either in enabling the functionality of the remote agent, or to voluntarily use and process the modified SOAP headers. Our future work includes the development of an API that facilitates the implementation of Web services which allow for this level of logging.

Concerning orchestration log collecting, we can propose an alternative solution. Since the most web services orchestration are using a BPEL engine, which coordinates the various orchestration’s web services, interprets and executes the grammar describing the control logic, we can extend this engine with a sniffer that captures orchestration information, i.e., the orchestration-ID and its instance-ID. This solution was possible thanks to the existence of the orchestration engine, it provides a centralized solution, but a simpler and less constrained solution than the previous one which collects choreography information.

Related Web services mining levels: Using these developed logging facilities, we aims to take into account web services’ neighbours in the mining process. The term neighbours refers to other Web services that the examined Web Service interacts with. The concerned levels deal with mining web service choreography interface (abstract process) through which it communicates with others web services to accomplish a choreography (i.e. level 2), or discovering the set of interactions exchanged within the context of a given choreography or composition (i.e. level 3 and level 4). The result of the mining efforts could be an interaction graph for each Web Service and its neighbours with the assumption of the existence of a (declarative) “composition language”. A web service may have multiple interaction graphs. The graph should contain information about what type of interaction links the Web Service to a specific neighbour and from where in the Web Service the call is coming from. Each one of these interactions graphs is related to a choreography or an orchestration and describes related web service behaviour.

This mining level can be seen as an extension to the BPEL or WS-CDL, as contribution provided by a third party to allow the mining and monitoring of Web Service. Several mining techniques proposed in workflow mining process context could be applied.

3. Conclusion

This paper presents a framework of our future works. After identifying the mining levels and defining the logging solutions, we are working to exploit existing process mining techniques [14,15] over captured Web services logs. Basically, we purpose to benefit from the results and findings of researchers in the field of process mining, for especially the web services mining highest level to enhance Web services composition and execution reliability.
4. Bibliography


