Discovering web service workflows using web services interaction mining

Schahram Dustdar,* and Robert Gombotz

Vita Lab, Distributed Systems Group,
Institute of Information Systems,
Vienna University of Technology, Vienna, Austria
E-mail: dustdar@infosys.tuwien.ac.at
*Corresponding author

Abstract: Recovering and analysis of workflows executed in Service-Oriented Systems (SOS) can be expected to gain more and more relevance in service-oriented information technology in the not so distant future. Our work attempts to perform workflow mining in logs provided by service-oriented architectures. Firstly, we introduce the idea of Web Services Interaction Mining (WSIM) and give an overview of related fields of research. Secondly, we propose a classification of SOS according to the log information they provide and analyse mining opportunities in the respective classes. To illustrate the potential of WSIM, we describe a complete workflow mining cycle in a SOS of very rich log information. Since such rich logs are often not available, we then direct our attention to mining in limited logs. We present our first steps towards a workflow mining algorithm, specifically tailored to Web Services (WS). We conclude with a discussion of open issues and future work.

Keywords: Web Services (WS) mining; workflows; process mining.

Reference to this paper should be made as follows: Dustdar, S. and Gombotz, R. (2006) 'Discovering web service workflows using web services interaction mining', *Int. J. Business Process Integration and Management*, Vol. 1, No. 4, pp.256–266.

Biographical notes: Schahram Dustdar is a Full Professor at the Distributed Systems Group, Information Systems Institute, Vienna University of Technology (TU Wien) where he is the director of the Vita Lab and Honorary Professor of Information Systems in the Department of Computing Science at the University of Groningen (RuG), The Netherlands.

Robert Gombotz is researcher and PhD student at the Distributed Systems Group.

1 Introduction

With the emergence of Web Services (WS) as a widely accepted standard, Service-Oriented Architectures (SOA) increasingly gain attention, both, in research and in industry. The use of WS facilitates the integration of formerly independent systems. In the future, service-oriented systems can be expected to gain more and more importance in Information Technology (IT). Designing new systems based on the service-oriented paradigm allows for the development and the deployment of loosely coupled systems which promise to be more flexible and more easily adaptable when business process requirements change over time. Such quickly evolving systems demand for sophisticated and powerful monitoring in order to ensure correct system behaviour and to allow for optimising system characteristics, like performance or usability. In other fields, mining techniques are applied to gain additional knowledge about systems and data. In this paper, we argue that mining may also be applied to service-oriented systems.

In our previous work we have introduced the idea of Web Services Interaction Mining (WSIM) (Gombotz and Dustdar, 2005; Gombotz et al., 2005). In WSIM we make use of the findings in the fields of data mining and process mining and apply them to the world of WS and SOA. Mining describes the act of examining and analysing large amounts of (log) data with the ultimate goal of Knowledge Discovery (KD). Consequently WSIM attempts to apply mining techniques on logs provided by WS and SOA infrastructures. We begin by presenting an overview of WSIM to outline the cornerstones of our approach.

1.1 Fundamentals of WSIM

We currently develop our WSIM approach with regards to three levels of abstraction that represent three complementary 'views' on WS.

The lowest level of abstraction is the *web service operational* level. On this level only one web service is to be examined. Its standing within a service-oriented system is not yet considered. Characteristics of a WS to be examined on this level include, but are not limited to, execution time, service usage or service availability. Mining on this level can be as detailed as to only analyse single operations of a given WS.

The second level of abstraction is the *web service interactions* level. On this level the focus is still on a single web service but with regards to its interactions with other services. In these interactions the given WS may act as either provider or consumer. Therefore, analyses on this level

may deal with WS conversation, that is, in what order are operations invoked by service requestors or WS composition, that is, which services are consumed by a given composite WS (Alonso et al., 2004). Furthermore, a thorough analysis of past interaction with a given WS can reveal dependencies in service-oriented systems and provide the information necessary for an impact analysis when changes to a WS are planned. The result of mining on this level can be an interaction graph as presented by Gombotz et al. (2004).

The highest level of abstraction is the *WS workflow* level. Here, we attempt to identify workflows or (business) processes, that are executed using the functionalities of WS. In that sense WSIM is positioned one step before WS orchestration. WS orchestration allows for the definition of workflows in an orchestration language such as WSBPEL (OASIS, 2004) and for a monitored execution of that workflow using a BPEL engine (Active BPEL, 2004; Oracle, 2004). However, WSIM deals with service-oriented systems that do not yet apply WS orchestration tools. A future application of WSIM on the workflow level might therefore be to generate abstract workflow definitions of discovered processes and thereby facilitate the work of workflow designers.

This paper focuses on the discovery of WS workflows. We want to emphasise that we are working in environments where no means of WS orchestration, for example BPEL, are used.

The rest of this paper is structured as follows. In Section 2 we examine other fields of mining and discuss their relevance for workflow mining in SOA. In Section 3 we analyse logging possibilities in SOA and propose a scale of levels of logging as well as the mining opportunities on these levels. Section 4 presents an example of WS workflow mining in a system of level 5 logging according to the scale proposed in Section 3. In Section 5 we move to service-oriented systems that provide less logging information and discuss the challenges of process mining in such systems. Section 6 presents our proposed approach to mining of WS workflows. Since our proposed algorithm is still work in progress, there are some problems and open issues which are discussed in Section 7. A conclusion is given in Section 8.

2 Related work

In this section we present the current state-of-the-art in other fields of research which have influence on WSIM, the most relevant of which are Web Usage Mining (WUM) and process mining.

2.1 Web usage mining

With the ever-growing importance of the World Wide Web (WWW) many researchers have directed their attention to developing means of managing, analysing and understanding the vast amounts of data provided on and by the web. These efforts are centred around applying data mining strategies to the content of web sites, web server log records, etc. Depending on the data that is analysed and the desired outcomes of these analyses, web mining can be further broken down into subcategories: *Web content mining*, *Web structure mining* and *Web usage mining* (Patricio Galeas, 2005). WUM and possibly web content mining provide methods and approaches that can be integrated and used in WSIM.

WUM is concerned with discovering web access patterns in web server logs in order to analyse user browsing behaviour (hyoKNOWsis, 2004; Wu et al., 1998). Most web servers provide logging features, which record one log entry for each request received by the server. Such a log entry typically contains the following information (Apache Software Foundation, 2005): the IP-address of the requesting host, a timestamp, the request line, for example, `'GET /index.html HTTP/1.0'`, and the HTTP status code returned to the client. Another important element a log entry may contain is the *referer* (sic). This is an identifier a client may include in his request indicating where he was referred from when requesting this resource, that is, where the link was he clicked to request the current resource. Also, if user authentication is required to access a website, the username is included in all log records of requests sent by that user.

In order to discover access patterns it is first necessary to group individual requests into user sessions, a process called session reconstruction. A session describes a user's visit to a website from opening the first page until the site is left. Once sessions have been identified, WUM mines in these sessions for reoccurring patterns. The knowledge gained is used to optimise or customise websites. Session reconstruction can be done by collecting requests sent from a given host and applying time constraints (Spiliopoulou et al., 2003). Such time-oriented heuristics pose limitations on the duration of the entire session (h1) and on the duration of a stay on one individual page (h2). The values for h1 and h2 are often fixed, for example, to 30 and 10 min, respectively. Some argue that these values should be flexible and should also incorporate the structure and content of a website (Berendt et al., 2002; Zhang and Ghorbani, 2004).

2.2 Process mining

Workflow mining or *process mining* (the terms may be used interchangeably), describes the effort to discover workflow patterns in a given set of log data. A workflow is a reoccurring, ordered set of *activities* that are performed in order to fulfil a higher-level task. Each individual execution of that workflow is called an *instance* of the workflow, or a *case*. The goal of process mining is to analyse a so-called *workflow execution log* in order to construct a *workflow model* that best describes *all* recorded instances of that workflow. Log entries must include a workflow identifier and a case identifier, so that the recorded events can be mapped to the according workflow and case.

The greatest challenge in process mining is the construction of models for complex processes. Workflows may contain alternative flows, concurrencies or loops (van der Aalst, and Weijters, 2003). Such complex patterns pose challenges to the development of efficient mining algorithms. In van der Aalst et al. (2003) present the current state of process mining. In Herbst and Karagiannis (2003) the InWoLvE process mining system is presented which claims to be able to discover a wide range of process models. In Schimm (2003) an algorithm for 'mining exact models

of concurrent workflows' is introduced. More issues are discussed by van der Aalst and Weijters (2003).

2.3 Applicability of WUM and process mining in WSIM

Web usage mining will have an impact on our work of developing methods to discover workflows in WS-related logs. Comparable to the browsing through websites, web service interactions can also be seen as isolated request-response transactions, embedded in larger-scale sessions made up of numerous such transactions. The records one can keep of WS interactions are somewhat similar to records found in web server logs. Especially the scenarios addressed in Zhang and Ghorbani (2004) and Wu et al. (1998), that is, mining in web server log records in the absence of session information, pose the same problems as the situation we are faced with in WSIM. However, the values used in time-oriented algorithms for user session reconstruction, namely h1 and h2, may be suitable in human user interaction scenarios, where page requests often do happen in intervals of less than 1 min. However, in service-oriented systems which are designed for machine-machine interactions, the time elapsed between two interactions may be far less. Therefore, new metrics will have to be found when applying WUM approaches in WSIM. Also, the structure of the system to be examined will have an impact of the values assumed for h1 and h2, which is also one of the key statements in Zhang and Ghorbani (2004). For example, in a system that performs a workflow of ordering products, time heuristics similar to those in web usage mining may be suitable. On the other hand, in a system for chemical simulations, in which certain services compute parts of computationally intensive equations, interactions might happen in intervals of only a few milliseconds. Therefore, there are limitations to the possibilities of applying WUM in service-oriented systems.

Another disadvantage in the approaches of web usage mining that itdoes not go as far as attempting to construct complete process models. The construction of such a model, however, is the main goal of WSIM on the workflow level. This goal is clearly shared with process mining. A major advantage of process mining is the availability of powerful algorithms, which can clearly be of use in WSIM research. A draw-back of process mining is, in our opinion, the restrictive assumptions made about the richness of information available in workflow execution logs (van der Aalst et al., 2003).

From our research we conclude that WSIM is located between WUM and process mining. WUM attempts to find frequent traversal path patterns from a limited amount of log information. Process mining, on the other hand, tries to construct complete workflow models from very rich log data. The log data available in WSIM is comparable to that in web server logs. However, the more steps are taken and the earlier WSIM is considered in the development process of a service-oriented system, the richer the log information can become. Therefore, WSIM on the workflow level should both

(a) apply web usage mining techniques on service-oriented systems that provide little log information and
(b) present methods for the development of more sophisticated logging in service-oriented systems, so that these systems may later be mined for exact workflow models with the assistance of process mining tools.

Consequently, the quality of workflow models that can be discovered using WSIM will depend on the quality and richness of execution log data provided by the underlying system.

3 Web services loggging

In this section we examine and formalise the logging possibilities in SOA. 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 respective features.

3.1 Level 1: Standard HTTP-server logging

The most commonly used log formats provided by web servers are the *Common Log Format* and the *Combined Log Format* (Apache Software Foundation, 2004b, 2005). The log entry recorded in Apache Tomcat when a request is sent to a WS ExampleService may look as follows:

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

The log entry contains the requestor's IP address, a timestamp, the request line, the HTTP code returned by the server, that is, 200 for OK, the size of the returned resource and the User-Agent, that is, Axis/1.1. The empty element, that is, '-', indicates that no referer-information is available. Such log records allow for tracking of the service consumer, determining which service is called how often (but not which operation of the service) or analysing service failure rates.

This level of logging can be achieved by simply enabling the respective web server's logging facilities. The service-oriented environment is in no way affected.

3.2 Level 2: logging of complete HTTP requests and responses

Alternatively to web server logging, an HTTP listener may be used to record all traffic directed to a given port, which allows for logging of the complete HTTP request and response traffic. This way the involved SOAP messages are also recorded since they are sent as part of the HTTP messages.

Achieving logging on level 2 requires an HTTP level logger that listens to a given port A and redirects to another port B. Also, it is necessary to reconfigure the HTTP server from the original port A, where service calls are expected to be received, to port B, to which the HTTP logger redirects. The service-oriented system is not affected. An existing open-source HTTP listener is Apache TCP Tunnel/Monitor (Apache Software Foundation, 2004a).

3.3 Level 3: logging at WS container level

On level 3 the logging is done inside the WS container, for example, Apache Axis (Apache Software Foundation, 2004c). Unlike on levels 1 and 2, the logging is more flexible and can be configured, for example, by only monitoring and logging certain WS. Also, HTTP requests not directed at WS are ignored. The logging itself is achieved using SOAP intermediaries. In Apache Axis such intermediaries are called *handlers* and they can be added to each deployed web service's request and/or response flow.

This level of logging can be reached by developing SOAP intermediaries that log all SOAP traffic and integrating them into the service-oriented system. The WS themselves are not affected.

3.4 Level 4: logging client activity

Logging on levels 1 through 3 involves provider-side logging only. Such scenarios are comparable to the situations addressed by web usage mining where interaction with a single, central server is assumed. On level 4 logging should also be done on the consumer side, for example, when a WS is a composite service and itself makes calls to other WS possibly deployed on other hosts. Level 4 logging, therefore, greatly expands the opportunities of workflow mining, since a system's activities as a client are also recorded. If a workflow spans over multiple hosts or systems, such interactions may be discovered if level logs are available.

Apache Axis allows for the use of client-side handlers which are invoked whenever the Axis API is used to consume services (Apache Software Foundation, 2004c). The system itself is not affected when level 4 logging is to be implemented.

In Serra et al. (2004) the authors present a sophisticated logging architecture for service-oriented systems. Their logging facilities reach level 4 logging on the scale we propose. Some of the authors' goals they wish to reach using the log architecture are also addressed by WSIM on the operational level, such as service execution time and web service usage.

3.5 Level 5: providing for process information

As stated in Subsection 2.3, successful mining for exact workflow models requires workflow information in log records. We have also shown that such information is not available in conventional web server logs. One of the goals of WSIM is to make service-oriented systems fit for process mining by providing suitable logs.

Therefore, the highest level of logging in SOA must provide for both a workflow 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, WS must 'cooperate' in a sense that they are able to receive and forward information regarding the workflow they are currently part of. The exchange of that information between WS can be done by using SOAP headers. WS should process the workflow information and forward it to other WS it consumes in the process. If such measures are taken when designing WS the recorded logs will allow the mining of exact models of complex workflows. Our future work includes the development of an API that facilitates the implementation of WS which allow for level 5 logging Table 1.

Table 1 Summary of logging features and mining opportunities

L	*Logged information*	*Logging facility*	*Mining opportunities*
1	– consumer IP – invoked WS – timestamp – HTTP status code	– Web server	– service utilisation – consumer tracking – failure rates
2	– level 1 + – SOAP request & response – timestamps	– HTTP listener & logger	– level 1 + – WS execution time
3	– invoked WS & operation – SOAP request & response – timestamps	– WS container – SOAP handlers	– level 2
4	– level 3 + – consumer-side activity	– WS container – SOAP handlers	– level 3 + – client-side activity
5	– level 4 + – workflow information	– level 4 + – Web services	– level 4 + – WS workflows

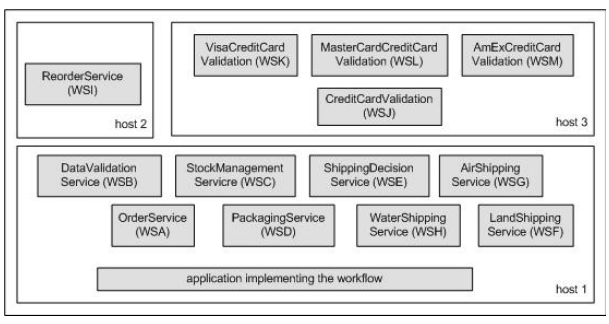
4 Mining on Level 5

In this section we present a motivating example showing possible applications of WSIM. Although only simple, the example has been fully implemented and it demonstrates, on a small scale, what could be done with WSIM on a much larger scale in the not so distant future. We first present the scenario and then describe the mining process.

4.1 Motivating example

The sample service-oriented system we have developed is a collection of 13 WS which are shown in Figure 1. We assume to have ownership over host 1, that is, access to its logs, while hosts 2 and 3 are third-party owned.

Figure 1 Case study service oriented system



The workflow is started by a client application making a call to the OrderService (WSA), which triggers the execution of an application implementing the workflow by using the other WS. The DataValidationService (WSB) is called to check if the customer is registered and if the given product is available in the desired quantity. If the product is not available the StockManagementService (WSC) is called, which in turn makes a call to the ReorderService (WSI) on host 2. Furthermore, the DataValidationService (WSB) calls a CreditCardValidation (WSJ) on host 3. The CreditCardValidation-WS makes, depending on the company in the credit card information supplied, a call to either the Visa-, the MasterCard- or the AmEx-web service. Once order data and customer data have been successfully validated, the PackagingService (WSD) is invoked. When completed, the ShippingDecisionService (WSE) is called which determines the method of shipping and finally invokes one of the ShippingServices. This completes the workflow.

In addition to the example workflow we have implemented logging facilities which reach level 5 on our proposed scale, that is, the WS are able to receive, process, and forward workflow information. The SOAP enabling software used was Apache Axis (Apache Software Foundation, 2004c), deployed on an Apache Tomcat servlet engine (Apache Software Foundation, 2004b). The logging is done by SOAP handlers on both the client and the server side which are invoked before and after the invocation of each WS. Therefore, all request and response messages are logged. Each log record is of the format

```
uniqueID - mappingID- targetWS - msgType -
          SOAPmessage - timestamp
```

where mappingID is an identifier mapping a SOAP request to its corresponding response, targetWS is the URL of the invoked service, msgType is the type of message, that is, request or response and SOAPmessage is the complete SOAP message that was sent.

4.2 The mining process

In our experiments the workflow was executed several times so that every possible flow of the process was performed at least once, for example, in at least one instance of the workflow products were unavailable and the ReorderService had to be called. Once a sufficient amount of data was collected the log records were preprocessed to a format more suitable for data mining. For example, the recorded SOAP messages were analysed for workflow information in their headers and the invoked method was retrieved. Also, corresponding log records, that is, request and corresponding response, were combined into one log record, so that one call to a web service was represented by one record. The log format after data preprocessing was as follows:

```
  workflowID - instanceID - targetWS -
operation - SOAPrequest -
  requestTimestamp - SOAPresponse -
responseTimestamp
```

4.2.1 Scenario

In order to demonstrate the benefits of logging on level 5 consider the following scenario. The formerly independent hosts 1–3 in our case study are now under ownership of one virtual organisation. Such a situation might occur when companies merge or when the independent hosts were previously managed by separate departments and are now put under the control of one central IT-department or outsourced. In such a situation it is beneficial when workflow information is available for processes that are performed using services deployed on (formerly) independent hosts.

4.2.2 Data preprocessing for ProM

In our experiments we used ProM as the process mining tool (Technische Universiteit Eindhoven, 2004). ProM requires an input file in XML format which contains the information concerning processes and their instances. It should be noted that ProM works with event-based data in order to be able to discover complex workflow patterns. This means that not activities are recorded but rather events. The simplest events regarding an activity are `start` and `complete`. Therefore, the web service execution logs have to be transformed into an event-based view. The XML format of a ProM – input file is roughly the following:

The root element is the `<WorkflowLog>` element and it has a number of `<Process>` subelements, each encapsulating execution data of *one* process, or workflow. A `<Process>` element has a number of `<ProcessInstance>` child elements. A `<ProcessInstance>` has numerous `<AuditTrailEntry>` child elements. An `<AuditTrailEntry>` represents one log record and contains an identifier of the activity, the event-type and a timestamp.

The preprocessed logs described in the previous subsection were traversed for log records belonging to a given workflow, that is, ‘orderprocess’ in our case. Because ProM requires log entries to be sorted by process instances, a list of instances was constructed, which was then traversed until all instances were processed. As an activity identifier we used the values `targetWS-operation` of the preprocessed logs. Furthermore, the receiving of a SOAP request was taken as the `start` event of an acivity, the sending of the SOAP response was considered the `complete` event. Also, since both, provider- and consumer-side interactions were logged, care had to be taken that interactions were only considered once, in cases of provider and consumer residing on the same host.

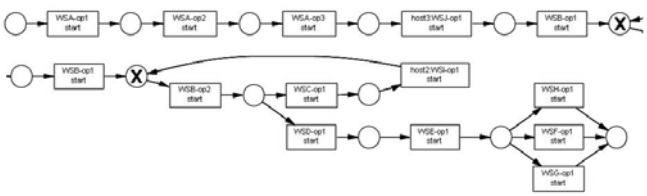
4.2.3 Mining results

The procedure of creating a ProM–compatible workflow log was performed several times with varying amounts of log data available. In one session only the logs collected at host 1 were used. This scenario may very well appear in the real world when system owners decide to upgrade their infrastructure to a BPEL–enabled environment.

The log data first had to be cleaned from interactions which were recorded twice, that is, by both provider-side and consumer-side handlers. For all interactions with third party hosts the consumer-side entries made at host 1 were used

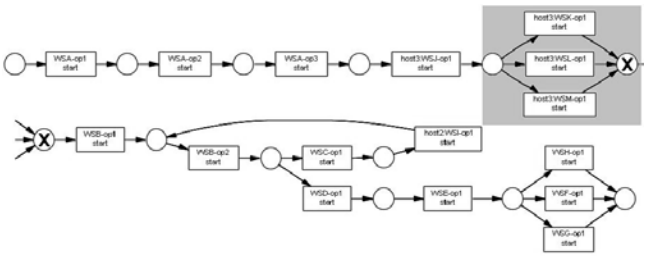
to capture these interactions. From the information extracted from the logs the workflow model in Figure 2 was constructed using ProM. The workflow model turned out to be complete and correct.

Figure 2 Process model of host 1



Note that the names of WS and operations were abbreviated in Figures 2 and 3 and only start-events were considered in order for the models to be displayable here. The abbreviations correspond to those used in Figure 1. Furthermore, the original images created by ProM had to be 'cut in half'. The circles marked X are where the two halves should be joined.

Figure 3 Process model of all hosts



In another experiment, we considered all logs collected on the three hosts. This scenario is imaginable when formerly independent hosts come into ownership of one virtual organisation. Again, the data was preprocessed and cleaned from double recorded interactions. The workflow model in Figure 3 was constructed which now includes the workflow performed on host 3. Figure 3 demonstrates the benefits from providing logs at a level 5 of our scale. The portion of the model highlighted is the additional knowledge gained from including logs from host 3 in the mining process.

5 Mining on lower levels

In the previous section we have presented an example of how WSIM can be performed in service-oriented environments where workflow information is available in seb service logs. Such environments belong to the logging level 5 category in the classification proposed in Section 3. Since such rich logs can, however, not be expected in most service-oriented systems, we now direct our attention to WSIM in systems of logging level 4 and lower. Such logs do not contain any workflow information, that is, we are looking at sets of indiviual interactions that are – to begin with – not related to each other. In the absence of workflow information, it is not possible to apply conventional process mining algorithms and tools as we did in Section 4. Other ways of and algorithms for discovering workflows need to be found and developed.

In the following, we present our first approach to the problem. In this approach we attempt to perform Session Reconstruction (SR) on WS logs somewhat analogously to session reconstruction in WUM. We argue that these sessions can also be seen as instances of workflows. It is from these instances that we will try to deduct workflows, like WUM deducts frequent web access patterns from individual sessions.

However, the nature of human user sessions can be very different from that of sessions in service-oriented systems which has a strong impact on our work of developing SR tools for WS logs. These differences are discussed in the following subsection.

5.1 User session reconstruction versus. SOA session reconstruction

As stated in the introduction, conventional (user) session reconstruction attempts to reconstruct human user web browsing sessions from web server logs in the absence of session information (Wu et al., 1998). This is usually done by placing global temporal constraints on the maximum time elapsed between individual page requests (h1) and on the maximum duration of the entire session (h2). Often used values for h1 and h2 are 10 min and 2 hr, respectively. In the process of SR a session is therefore running as long as 10 min are not exceeded between individual page hits and for a maximum of 2 hr. Such values are definitely applicable in human web browsing as they reflect often observed human browsing behaviour. However, improvements to this somewhat static approach have been suggested by Zhang and Ghorbani (2004) by using flexible and adjustable values for h1 and h2. In Berendt et al. (2002) the authors argue that the structure of a website should also have an impact on the temporal constraints used during session reconstruction.

While the choice of values for time constraints is (luckily) limited, this does not apply to service-oriented environments. Even though the exact values for h1 and h2 are arguable, they will always be in a certain range that is simply determined by human browsing behaviour. For example, even the most enduring web surfer will not browse a site for more than 5 hr in one sitting, nor will he be staying on one page for more than 2 hr. And even if such cases occur, they are so few in number that their impact on the analyses is neglibible.

The situation in service-oriented environments is different since interactions may be human-to-machine or machine-to-machine or even a mix of the two. Especially in machine–machine interaction it is difficult to place constraints on the duration of a session. Depending, of course, on the functionality and semantics of a given service-oriented system, sessions may last for much longer periods than in human-machine interactions. Also, a maximum value for the time elapsed between interactions with WS for them to be part of one session is difficult to determine.

5.2 Main problems of SR in SOA

This section discusses the major difficulties we face in SR in service-oriented systems. Some of them lie in the different nature of service-oriented sessions compared to human user sessions. In the following, the terms *User Session (US)* and *User Session Reconstruction (USR)* are used to specifically

denote sessions of human web browsing, whereas *Service Session (SS)* and *Service Session Reconstruction (SSR)* refer to sessions in service-oriented environments.

5.2.1 Diversity of possible sessions

USR deals with user sessions, which are relatively homogenious in their characteristics. In other words, all US have in common that it is a human user that is browsing through the pages of a website. Although websites and web pages differ in their nature and browsing behaviour is different on, for example, a news website than on a flight reservation site, it is generally accepted that US have a maximum duration and a maximum time lapse between page requests.

The situation in SSR is different. The number and variety of service-oriented applications that will emerge in the future are unpredictable as are the processes that will be executed in such environments. The possible diversity of service-oriented processes implies just as large a range of different SS that will be observed. Consider the following scenarios (Figure 4–6):

1) An application polls/consumes a web service every hour from Monday to Friday during working hours. On weekends or holidays the WS is not consumed. A graphical representation of the recorded interaction sequence is shown in Figure 4. The letters in the top row denote a service or more precise, a service operation. The bottom row specifies the time lapses between the service calls. The time elapsed between interactions is given in minutes.

2) An application allows a user to book a business trip to a city of his choice including flight, hotel, transfer from/to the airport and some spare-time activities, like a visit to the opera. The individual reservation transactions are implemented using WS provided by the respective companies. The time lapses between interactions in this case are roughly the same as in human user web browsing sessions.

3) An application for computationally intensive simulations makes calls to a number of WS whenever a simulation is executed and when additional computational resources are needed. Simulations occur unperiodically, the time lapses between service calls during a simulation are in the range of 10 sec to 30 min.

Figure 4 Interaction sequence example 1

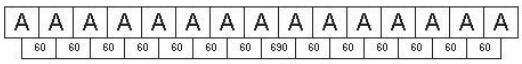


Figure 5 Interaction sequence example 2

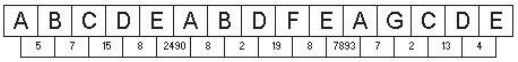


Figure 6 Interaction sequence Example 3

The above examples should clarify the different nature of US to SS and show the need for different time constraints in SSR than in USR.

5.2.2 Structure of sessions

Besides the different temporal nature of SS, SS may also be of a more complex structure than US. USs are purely sequential, meaning that indiviual requests in a session allways occur sequentially and never or concurrently. In SS, however, services may be executed concurrently and sessions may therefore fork and rejoin. This possibility adds complexity to session reconstruction in service-oriented systems. A detailed analysis of service interaction patterns can be found by Barros et al. (2004, 2005).

5.2.3 Cross-site sessions

A problem that is also observed but has not been solved in USR is that of *cross-site sessions*. The term denotes the case of a user browsing a given website A, leaving that site in the meantime to browse website B and then returning to website A. The user's interactions with website B are, obviously, not recorded in website A's server log and can therefore not be considered during session reconstruction. It can be argued that the use of relatively small values for h1, that is, the maximum time elapsed between page requests, limits the impact of such cases, since a session is simply considered to be completed when a user leaves website A for more than 10 min.

In SSR such scenarios are of greater impact to the session reconstruction process. Consider again example 2 earlier in this section (Figure 5). A user might request prices for airplane tickets in the beginning of his travel booking session. He might then query for hotels, request prices for rooms, decide to book a room at hotel X based on room rates and the location of the hotel and at the end of the session finally book his airplane ticket. The time elapsed between requesting the price for airplane tickets and actually buying the ticket may be anywhere between 30 min and, for example, 3 hr, a relatively high time lapse. Still, these two interactions are clearly part of one session. The question of whether such sessions can be discovered by a highly generic algorithm can only be answered by empirical studies.

When provided, WS-Addressing information may allow for grouping cross-site interactions into sessions, as is mentioned in Section 6.3.

5.2.4 Genericity of the algorithm

The problems described in the previous subsections are not unsolvable per se. If any of the given problems is a-priori known to exist in a given system, it can be addressed by specifically configuring the algorithm and the mining process. The great challenge is to develop an algorithm or a mining tool, that possibly recognises complex scenarios and delivers correct results, that is, that is generic and therefore applicable to a wide range of systems.

The question as to how generic our own approach is can only be answered after a sufficient amount of test runs have been performed. However, in order to reduce the complexity

of the mining, our algorithm allows – and encourages – to specify as much a-priori knowledge about the system as possible. Such knowledge may be a maximum duration of sessions, or any other variable used during SSR. More of these variables and factors are listed and explained in the following section.

6 Multiphase workflow discovery

6.1 Session reconstruction

The first step in the multiphase workflow discovery algorithm is Session Reconstruction (SR) which may be performed based on

a) temporal information or

b) the recorded interactions with WS.

Temporal information in logs is the intervals between service calls represented by the timestamps, *interactions* are the actual calls to WS. This paper only elaborates on temporal SR. SR based on WS calls is only mentioned for the sake of completeness.

6.1.1 Time-based SR

The temporal approach to SR builds upon SR in WUM. However, since time constraints from WUM are not automatically applicable in SOA such constraints should be calculated from the *interaction sequence* that is currently being analysed. The idea is to take the highest time-lapse value and to name it the 'session boundary threshold', that is, the time lapse that definitely marks the end of one and the beginning of another sessions. The session boundary threshold is then continually decreased until 'satisfying' results are achieved. (The term satisfying is, of course, relative and is further discussed later in this section). The higher the session boundary threshold may be kept the more reliable are the results of SR.

Consider interaction sequence 1 in Section 5.2.1 (Figure 4). In SR this interaction sequence or to be precise, the time lapses between interactions are analysed. In the given example the session boundaries, that is, the time lapses of 690, may be extracted easily since they deviate highly from all other values. This example also demonstrates why SR based on time intervals may in many cases be very efficient.

Furthermore, the performance of SR may be improved by defining maximum values for intervals between service calls to denote definite session boundaries. For example, a maximum time lapse of 24 hr may very well be justified in many service-oriented environments. Of course, depending on the nature and semantics of the system, even maximum time lapses of only 1 hr may make sense.

6.1.2 Interaction-based SR

Session reconstruction based on the recorded interactions is by far more dependent on the underlying service-oriented system. However, when applicable, SR of this type may be preferable to SR based on temporal values. Consider, for example, a system that requires authentication through a login-WS at the beginning of every session. In such a system sessions can be easily reconstructed from the recorded interaction sequences simply by starting a new session whenever an interaction with the login-service is found in the logs.

6.2 Testing quality of SR results

Once sessions have been identified using one of the previously described methods the quality of the results should be tested. Of course, there is no absolute measure of such quality. However, we suggest some criteria based on which the results of SR may be evaluated.

It is important to note that all the following tests may only *support* the tested results, that is, positive test results support the hypothesis that the results of SR are of high quality. Negative test results, however, should not be used to falsify SR results. This nature of the tests should become clearer in the following subsections.

6.2.1 Session similarity

One of our key arguments is that sessions of a given consumer will be similar to each other to a certain extent. This argument is based on the assumption that a service consumer has use for certain services – or workflows comprised of certain services – and has no use for others, just like a human web user is interested in certain pages or parts of a website and ignores others. Therefore, we argue that some similarity can be expected among sessions of a single consumer.

In order to measure similarity of sessions we propose the following criteria, which are listed in order of sophistication, beginning with the simpler criteria:

1. *Session duration*: two sessions of duration 20 and 30 (units of time) are more similar than two sessions of duration 20 and 60.
2. *Number of interactions*: a session that involved 5 interactions and a session that involved 8 interactions are more similar than two sessions of 5 and 15 interactions.
3. *Services consumed in a session*: two sessions [A:B:C] and [B:D:A] are more similar than two sessions [A:B:C] and [D:A:E]. Only the set of services used is considered.
4. *Number of service consumptions*: a refinement of the preceeding criterion. It is now also considered how often each service is consumed. Two sessions [A:B:A:B:C] and [A:C:B:A:B] are more similar than two sessions [A:B:A:B:C] and [A:B:C].
5. *Order of Services in a session*: two sessions [A:B:C:D] and [A:C:B:D] are more similar than two sessions [A:B:C:D] and [A:D:C:B].
6. *First (and last) service in a session*: this criterion is based on the assumption of entry and exit points of sessions/workflows. We argue that similar starting and/or ending servic(es) are an indicator for session similarity.
7. *Reoccurring patterns in sessions*: this criterion of similarity looks for reoccurring patterns in sessions

that are to be compared. By our definition, such patterns are subsequences of sessions that occur in all of the analysed sessions. For example, in the sessions [A:B:A:C:D:E] and [F:A:B:A:G:E:C:D] two reoccurring patterns can be discovered, that is, [A:B:A] and [C:D]. The more such patterns are found in the analysed sessions, the higher is the case for similarity. Furthermore, we argue that more complex patterns should be weighted higher when determining session similarity. That is, a reoccurring pattern of length 4 makes a higher case for similarity than a pattern of length 2.

This test should also take into account patterns that contain concurrencies and (possible) loops. For example, In the sessions [A:B:B:C] and [A:C:B:B:B] the sequences containing interactions with WS B should be considered a possible loop and [B:B] and [B:B:B] should be considered a reoccurring loop.

The session similarity tests can already clarify the non-falsifying nature of tests that was mentioned in the introductory paragraph of this section. A higher similarity among sessions found in SR does support the hypothesis that the sessions were identified correctly, although it is not proof of their correctness. On the other hand, low similarity among sessions does not automatically falsify SR results, since it is possible that a given consumer performs completely complementary sessions.

6.2.2 Parameter-based session tests

The parameter-based session test goes as far as analysing the content of SOAP-messages that were exchanged in interactions. Such an analysis is, of course, only possible in environments of logging level 2 and higher.

The idea is to collect SOAP messages that belong to one session identified in SR and analyse them for reoccurring input and/or output parameter values. Reoccurring parameter values support the hypothesis that a set of interactions belong to one session, for example, and order-ID that might be used throughout an individual order case. In other cases, an output parameter value of one service interaction might be used as an input parameter value in the following interaction.

Again, this test should not be used to falsify SR results. A session does not need to contain reoccurring parameter values but if it does it is a clear sign that individual interactions are part of a larger scale process or session.

6.2.3 Session similarity across users

The next step in testing SR results quality compares sessions of multiple users. This test takes the SR results of multiple users and compares them for similarity, somewhat analogously to the session similarity test. Again, the key argument is that similar sessions that have been discovered even for *multiple* users, and not just multiple times for a *single* user, are more probably actual sessions than sessions only performed by a single user.

The same restrictions apply to this test as to other tests, that is, the results may verify sessions but not falsify them.

6.3 WS-addressing

WS-Addressing is a proposed standard that was submitted to the W3C. It provides methods to address WS and messages in a transport-neutral manner by including addressing information in the SOAP envelope rather than leaving it to the transport layer, such as HTTP. Web Services Addressing (WS-Addressing) defines two interoperable constructs that convey information that is typically provided by transport protocols and messaging systems. The two constructs are *endpoint references* and *message information headers* (W3C, 2004). Message information headers are specifically interesting for our work.

These headers allow to assign a unique identifier to a given SOAP message, included in the `<message id>` element. This message-ID may later be referenced by other messages in a 'relates-to' manner. Such referencing of messages may be very useful or even necessary, in asynchronous communication or 1-to-*n* multilateral interactions.

It is fairly obvious how WS-Addressing information included in SOAP messages may be useful in mining for WS workflows. The ability to correlate messages allows a guaranteed grouping of messages that belong to the same workflow case, even in asynchronous or cross-site cases. Therefore, we may use WS-Addressing information to

a) verify SR results and
b) perform SR in multiparty workflows and asynchronous interactions.

Since WS-Addressing is a fairly new standard, it is not yet incorporated in many systems. Apache Axis2, however, already provides a reference implementation of WS-Addressing (Apache Software Foundation, 2006).

6.4 The case for testing SR results

Some might argue that tests which measure session similarity are not suitable for evaluating the quality of results of session reconstruction. After all, sessions do not *have* to be similar at all in order to be correctly identified sessions. This is undoubtedly true.

However, we have two arguments that support the testing of SR results: Firstly, we need some means of determining the quality of SR results. Otherwise we could never determine the quality of our algorithm. Secondly and more important, we do not perform session reconstruction only for the sake of discovering sessions. Our ultimate goal is that of workflow mining. It is therefore justified that we somehow filter out completely complementary sessions since they are of very little use in the construction of workflows.

6.5 Mining for workflows

The final step is the construction of workflows. This phase includes selecting sessions for mining, preprocessing the data and performing the actual mining using existing process mining tools.

6.5.1 Identifying cases of workflows

The challenge in this step is to determine which sessions may be considered cases of a mutual WS workflow. Although

this question may never be answered with absolute cartainty (see Section 7.6) we propose a set of tests that may identify corresponding sessions with a certain degree of reliability. These are some of the test already introduced in previous subsections.

Session similarity: sessions that belong to a mutual workflow should be similar to a certain degree. The factors for similarity are the same as those named in Subsection 6.2.1 although they are of different significance here. For example, session duration is of little relevance in this step.

We believe that the most important criterion is that of *reoccurring patterns* in sessions. Mutual subworkflows – which is just another term for reoccurring patterns – definitely support the hypothesis of two sessions being instances of one workflow.

The second important criterion is the *first and* to a lesser extent, the *last service* consumed in a session, again based on the assumption of entry and/or exit points of a workflow.

Other criteria, such as *services consumed in sessions* also apply. It should also be noted that session similarity whithin a *single* user's sessions is more important here than similarity across multiple users. This is based on the argument that sessions of a single user are per se more probably cases of a mutual workflow than sessions of different users.

6.6 Mining for workflows

Once a set of sessions is collected that are – with a certain probability – cases of one workflow, the recorded data needs to be prepared for mining, that is, the previously missing workflow information is a-posteriori injected into the data. In practice, this means assigning a case ID to each session and a workflow ID to the set of sessions. The resulting data set may then be used as input for conventional mining tools as we did in Section 4.

7 Known issues and problems

As mentioned before, the algorithm presented in the previous section is work in progress. Therefore, there are a number of open issues and known problems in WS workflow mining.

7.1 Sequentiality of analysis

We believe that it is important for a mining algorithm to be applicable also on portions of the data rather than on the entire dataset. In other words, the algorithm should deliver satisfying results even if only interactions recorded for example in the last two months are analysed and not all interactions from the past year. This quality of a mining algorithm decreases both the amount of data that is needed for analysis and the performance of the algorithm.

7.2 Interleaving sessions

A problem that is also observed in traditional WUM is that of interleaving sessions. In traditional web browsing such sessions occur when multiple web browsers interact with a web server through one single proxy. In such cases multiple sessions that happen at the same time appear to be one single session, since all requests come from one single IP-address. To our knowledge, there is no way of a-posteriori separation of such interleaving sessions.

Such cases may, of course, also occur in service-oriented SR, for example, if multiple applications running on one machine consume services from a given host concurrently. One possible solution to this problem is the use of WS-Addressing information in SOAP headers when available (see Section 6.3).

7.3 Inductive bias

A problem that is inherent to our SR algortihm is an inductive bias. Since the SR algorithm decreases the value of the session boundary threshold continuosly until sessions are found there is a danger of identifying incorrect sessions. However, incorrect sessions should be recognised by the session quality tests that are performed later. The inductive bias is also present when selecting sessions for workflow mining.

7.4 Measures of session quality

As stated before, there is still a need for a measure of session reconstruction results' quality. We have proposed some individual criteria for session similarity and introduced the parameter-based session test. However, an overall measure for SR result quality, either as an absolute point value or as a percentage value, is yet to be found.

We will also attempt to develop an overall measure for session similarity, which has a strong impact on the session quality measure.

7.5 Correctness of results

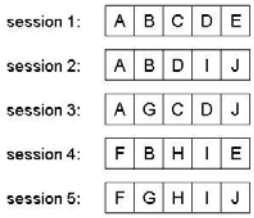
The correctness of session reconstruction results, that is, the identified sessions, can never be guaranteed. However, the more data they are backed up by, the higher the probability of correctness.

7.6 Session similarity in workflow mining

Before sessions are used in the actual workflow mining it is tested whether they could be instances of the same workflow. We believe this question can never be answered with 100% certainty. Consider two sessions [A:B:C:D:E] and [F:G:H:I:J]. At a first glance, one might never think of these sessions to be instances of a mutual workflow. However, when looking at Figure 7, which shows a set of recorded sessions, the situation changes.

Figure 7 shows the before mentioned sessions (sessions 1 and 5) as part of a set of sessions. When looking at the 'intermediate' sessions (sessions 2–4) the original sessions suddenly may be cases of a mutual workflow. They are simply the two most extreme cases of the same workflow.

Figure 7 Set of recorded sessions



8 Conclusion and future work

In this paper we have presented our approach of WSIM with a focus on mining for WS workflows. We have examined related work, of which we specifically point out WUM and process mining. Since the possibilities of process mining in SOA seem limited using traditional logging facilities like web server logs, we have presented a classification of service-oriented systems by the richness of the log data they provide. Depending on the level of logging, different methods of WSIM may be applied to a system. If level 5 logging is implemented, a service-oriented system may be mined for complete workflows using existing process mining tools such as ProM. As a motivating example we presented such a system and described the steps of processing the log data to create logs understandable by ProM. We also showed the results achieved when mining these logs for workflows.

Since level 5 logging cannot be assumed in many service-oriented systems, our current research is directed at developing mining algorithms for WS logs that do not contain workflow information. We have presented our first approach of extracting workflows from such logs using a multiphase algorithm. The algorithm is based on SR which is adapted for service-oriented environments. The identified sessions go through a number of tests that determine their usefullness in the construction of workflows. Sessions that seem to be cases of a WS workflow are collected and used in the actual mining for workflows.

In the near future we will test the proposed algorithm on data recorded by the service-oriented system that is currently being developed at our institute.

References

ActiveBPEL (2004) *ActiveBPEL Engine - Open Source BPEL Server* Available at: http://www.activebpel.org.

Alonso, G., Casati, F., Kuno, H. and Machiraju, V. (2004) *Web Services – Concepts, Architectures and Applications*, Springer-Verlag.

Apache Software Foundation (2005) *Log Files - Apache HTTP Server*, Available at: http://httpd.apache.org/docs-2.0/logs.html.

Apache Software Foundation (2004a) *Apache SOAP Documentation: User's Guide*, Available at: http://ws.apache.org/soap/docs/.

Apache Software Foundation (2004b) *The Jakarta Site - Apache Jakarta Tomcat*, Available at: http://jakarta.apache.org/tomcat/.

Apache Software Foundation (2004c) *WebServices – Axis*, Available at: http://ws.apache.org/axis/.

Apache Software Foundation (2006) *Axis 2.0*, Available at: http://ws.apache.org/axis2/.

Barros, A., Dumas, M. and ter Hofstede, A. (2004) *Service Interaction Patterns*, Available at: http://www.serviceinteraction.com.

Barros, A., Dumas, M. and ter Hofstede, A. (2005) 'Service interaction patterns', *Proceedings of the Third International Conference on Business Process Management*, Nancy, France, September, Springer Verlag.

Berendt, B., Mobasher, B., Nakagawa, B. and Spiliopoulou, M. (2002) 'The impact of site structure and user environment on session reconstruction in web usage analysis'. *Proceedings of the 4th WebKDD 2002 Workshop at the ACM-SIGKDD Conference on Knowledge Discovery in Databases*.

Gombotz, R., Dustdar, S. and Baina, K. (2004) 'Web services interaction mining', *Technical Report TUV-1841-2004-16*, Vienna University of Technology, Available at: http://www.infosys.tuwien.ac.at/Staff/sd/papers/TUV-1841-2004-16.pdf.

Gombotz, R. and Dustdar, S. (2005) 'On web services workflow mining', *Business Process Management Workshops*, Nancy, France, September, Volume 3812 of LNCS, Springer Verlag, pp. 216–229.

Gombotz, R., Baina, K. and Dustdar, S. (2005) 'Towards web services interaction mining architecture for e-commerce applications analysis', *Proceedings of the Conference on E-Business and E-Learning*.

Herbst, J. and Karagiannis, D. (2001) 'Workflow mining with InWoLvE', *Computers in Industry*, Vol. 53, pp. 245–265

hypKNOWsis (2004) *hypKNOWsis ... Making Sense of Your Data*, Available at: http://www.hypknowsys.org.

OASIS (2004) *OASIS Web Services Business Process Execution Language (WSBPEL) TC*, Available at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel.

Oracle (2004) *Oracle BPEL Process Manager*, Available at: http://www.oracle.com/technology/products/ias/bpel/index.html.

Patricio Galeas (2005) *Web Mining*, Available at: http://www.galeas.de/webmining.html.

Schimm, G. (2003) 'Mining exact models of concurrent workflows', *Computers in Industry*, Vol. 53, pp. 265–281.

Serra, S., Campos, M., Pires, P. and Campos, L. (2004) 'Monitoring E-Business web services usage through a log based architecture', *Proceedings of the IEEE Interantional Conference on Web Services (ICWS'04)*.

Spiliopoulou, M., Mobasher, B., Berendt, B. and Nakagawa, M. (2003) 'A framework for the evaluation of session reconstruction heuristics in web usage analysis', *INFORMS Journal of Computing, Special Issue on Mining Web-Based Data for E-Business Applications*.

Technische Universiteit Eindhoven (2004) *Process Mining Reserach*, Available at: http://www.processmining.org.

van der Aalst, W. and Weijters, A. (2003) 'Process mining: a research agenda', *Computers in Industry*, Vol. 53, pp. 245–264.

van der Aalst, W., van Dongen, B., Herbst, J., Maruster, L., Schimm, G. and Weijters, A. (2003) 'Workflow mining: a survey of issues and approaches', *Data and Knowledge Engineering*, Vol. 47, pp. 237–267.

W3C (2004) *Web Services Addressing (WS-Addressing)*, Available at: http://www.w3.org/Submission/ws-addressing/.

Wu, K., Yu, P. and Ballman, A. (1998) 'SpeedTracer: a web usage mining and analysis tool', *IBM Systems Journal*, Vol. 37.

Zhang, J. and Ghorbani, A.A. (2004) 'The reconstruction of user sessions from a server log using improved time-oriented heuristics', *Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)*.