Probing and Monitoring of WSBPEL Processes with Web Services

Heinz Roth, Josef Schiefer, Alexander Schatten
Institute for Software Technology and Interactive Systems
{hroth, js, aschatt}@ifs.tuwien.ac.at

Abstract

Today's business climate requires organizations to constantly evolve IT strategies to respond to new opportunities or threats. Tracking the achievement of business goals, objectives and strategies is increasingly used to measure and adjust the outcome of business processes. In this paper, we introduce a web service based approach for probing WSBPEL processes. With our approach organizations are able to automatically extend existing WSBPEL processes with auditing extensions which capture audit information during process execution time. We show how to transform a WSBPEL model into an auditable model which can be used for process monitoring purposes. Based on our experience on building an auditable WSBPEL model we propose some extensions to the WSBPEL specification.

1. Introduction

Business Process Management Systems (BPMSs) are software solutions that support the management of the lifecycle of a business process. This includes the definition, execution and monitoring of business processes. For the execution of business processes, many organizations are increasingly using process engines supporting standard-based process models (such as WSBPEL [3]) to improve the efficiency of their processes.

A major challenge of current BPMS solutions is to close the gap between executing a business process and monitoring it. Sayal et al. state in [13] that this combination, among other things, allows business users to be notified or even react in real-time in case predefined conditions are met. Furthermore, many types of events may be recorded during the execution of the business process, including the start and completion time of each activity, its input and output data, the assigned resources, and the outcome of the execution. This information can be used for analysis purposes in order to reveal problems and inefficiencies in process executions and identify solutions in order to improve process execution quality. In addition, information on active processes can be used for notifying users and processes of quality degradations.

There exist plenty of WSBPEL engines, commercial and open source ones, which offer their own auditing features. Unfortunately most of these proprietary auditing features are not designed for interoperability with other WSBPEL engines, reducing their value as a monitoring instrument not only for inter-organizational processes where heterogeneous systems are most likely to be found.

One reason for the limited capabilities is the lack of a broadly supported (industry) standard for audit trail information, which is implemented by major BPMSs. Although there is a standard specification for the workflow audit trail in the reference model of the Workflow Management Coalition [14], it is not support by most workflow management products. Therefore, up till recently, it has been very difficult for process analysts to capture and use process audit information to get a clear picture regarding the status and performance of business processes.

In this paper, we propose a new approach to monitor WSBPEL processes which overcomes the mentioned problems. The basic idea behind our approach is to extend a WSBPEL process definition with auditing activities in order to report interesting state changes to an auditing web service. The resulting auditable process definition does not use proprietary elements but remains compliant with the WSBPEL standard. Therefore, it is possible to enhance process definitions running on different WSBPEL engines with the proposed auditing feature and centralize the collected information.

The remainder of the paper is organized as follows. In Section 2 we reflect on related work on auditing and monitoring business processes. In Sections 3 - 5 we propose a new approach for

transforming an existing WSBPEL process description into an auditable version. In Section 6, we propose auditing extensions for the WSBPEL specification and finally, in Section 7 we conclude our paper and give an outlook for future work.

2. Related Work

For the execution and monitoring of business processes, many organizations are increasingly using BPMSs to improve the efficiency of their processes and reduce costs. During the execution of the business process, workflow management systems record many types of events, such as the start and completion time of each activity, the assigned resources, and the outcome of the execution. Major BPMSs lack capabilities for transforming, accumulating and condensing audit trails of distributed business processes and using this information for monitoring and analysis purposes to provide feedback about the performance of business processes (in particular inter-organizational business processes).

Leymann and Roller point out that audit trail data from automated business processes can quickly increase to a sizable amount [8]. If several organizations are involved, e.g. if processes across an integrated supply chain are to be supervised, the volume of information can quickly reach levels that negatively impact the operational performance of enterprise applications. One way to handle data intensive operations is the outsourcing of data processing to external parties. The outsourcing of process management services allows for the sharing of process information across multiple organizations. The positive economic effect of shared information in an integrated supply chain has been demonstrated in a number of studies [4][5].

Jeng and Schiefer have developed an agent-based architecture with the aim of providing continuous, real-time analytics for business processes [6]. For the analytical processing they introduce an agent framework that is able to detect situations and exceptions in a business environment, perform complex analytical tasks and reflect on the gap between current situations and desired management goals. McGregor and Kumaran have presented a Solution Management framework that analyzes workflow audit logs, utilizing decision support system principles and agent technologies to feedback performance measures [9]. This framework forms part of the Intelligent Workflow Monitoring System (IW-MONS) meta-methodology. An extension for this framework using web services was proposed by McGregor and Schiefer, who state that current web service frameworks do not include the functionality required for web service execution performance measurement from an organizational perspective [10]. Part of this work is the extension of the Business Process Execution Language for Web Services (BPEL4WS) with mechanisms to obtain performance information. Zur Muehlen has developed a generic architecture for workflow-based Process Information Systems [11]. Based on the flow of management information in an organization, he developed a cybernetic model of three distinct feedback levels for automated process regulation, operational process management, and strategic process management.

Baresi and Guinea [1] as well as Lazovik, Aiello and Papazoglou [7] state that correctness and quality of applications in the service-oriented world, which are captured in an increasingly abstract manner, cannot be assessed with the same techniques as other software pieces. Underlying services and their providers as well as the process structure itself can be adjusted very easily which is one of the strengths of the SOA paradigm, but it has the disadvantage of a never-ending testing deficit. Therefore, particularly Baresi and Guinea propose that the validation of these systems has to be shifted to the run-time. Similar to our approach, they extend existing WSBPEL processes to contact a special web service called monitoring manager. But unlike our proposal, their goal is not to monitor process instances to gain real-time information which can be used to assist managerial tasks but to detect participating services which deliver unexpected results.

3. Auditing WSBPEL Processes

In this section, we introduce various strategies for auditing WSBPEL processes; there are principally five options: 1) instrumentation of web service requests of the WSBPEL process on protocol level and 2) on application server level, 3) utilizing the auditing service of a process engine used for enacting the WSBPEL process, 4) using probes in the operational systems that track state changes of the business process, or 5) include the auditing mechanism as a partner within the WSBPEL process,. All five options have advantages and disadvantages and are discussed in the next sections.

3.1. Inception of Web Service Requests on (HTTP) Protocol Level

Using this option, a web service gateway is used to intercept any web service request of the BPMS and extract the data needed for the auditing. The major advantage of this option is that the web service auditing is fairly straightforward and many tools are available that allow an interception of web service requests (e.g. IBM Web Service Gateway). However, the web service requests of WSBPEL processes have a limited visibility to internal process information. For instance, the web service requests do not include details about the process instances, or variables used by the process engine to control the process execution. Therefore, with this option only an incomplete audit trail can be extracted which often results in a more complex audit trail processing in order to regain further process information.

3.2. Inception of Web Service Requests on Application Server Level

Web services are usually executed on application servers which offer facilities for listening and incepting client requests. For instance, J2EE and .NET offer an event and listener models for capturing audit data from web requests. Developers can take advantages of such facilities and implement their own listener components which extract relevant information form web service requests. Although, this approach is often easier to implement and deploy than using a tool for the interception of web service requests on HTTP protocol level, it has the same deficiencies when it comes to the visibility of internal process information.

3.3. Auditing Service of BPMSs

Most BPMSs supporting WSBPEL provide auditing services that record the activities of the process engine. However, the auditing services can vary significantly between BPMSs. Although the Workflow Management Coalition has already defined standards for workflow logs [14], the major vendors of BPMSs still mostly use proprietary audit trails. However, using the auditing services of BPMSs directly has also some advantages over the previously discussed options. If we are able to directly instrument WSBPEL flows in a process engine, we have full access to all runtime data and thereby, we can extract more contextual information about the business process. This may include state information about the process instance or information about the assigned resources to an activity.

Such an auditing service has full access to the runtime information of a business process and can decide during process execution which runtime data has to be extracted. An observer plug-in (e.g., in IBM Websphere) can be used by developers to directly forward relevant auditing data.

Although, this option is the most flexible one for extracting and propagating audit trail data with minimal latency, only few process engines support this type of flexible auditing. Another trade-off for this approach is that it is proprietary to BPMSs.

3.4. Probes in Operational Systems

In order to track state changes of business processes also probes in the operational systems can be used. Probes can sense the operations of a source system and also extract and generate audit data from the operational run-time data. They can be implemented at various application layers (e.g. database layer, business-logic layer, presentation layer), but must be exposed to sufficient runtime information about the business process in order to perform the logging activities.

For instance, triggers on database tables, can automatically generate audit data each time a table record is inserted. Although the delays for extracting and propagating the audit data are minimal, database triggers have a performance impact on the operational source system and sometimes don't have easy access to all run-time data of a business process (e.g. data of business objects has to be gathered from multiple tables).

Often the operational source systems offer auditing mechanisms that can be utilized. By using logging daemons that extract information from log files or database tables on a scheduled basis, the required audit data about the business process can be generated. Due to the scheduled data extraction, this option causes some latency for the audit trail processing.

Figure 1 shows a business processes supported by four operational systems with three of them providing probes. An auditing service collects the logging data from various probes and consolidates it for further use. The biggest challenge with such an approach is the last step. The consolidation and integration of data from probes of various source systems is a major undertaking, since the formats, semantics and quality of the audit data can vary significantly for the source systems.

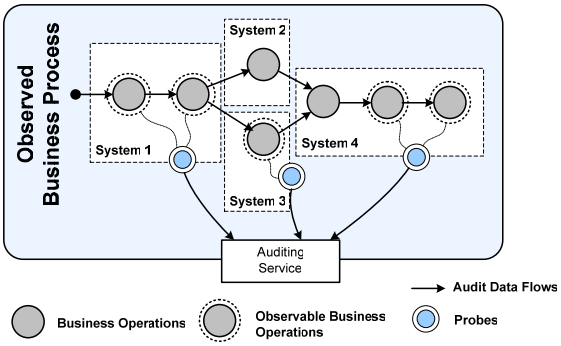

Figure 1: Operational systems with probes

3.5. Auditing Web Service as Part of a WSBPEL Process

An *auditing web service* provides the mechanism to gather audit trail information about business processes during execution. The captured audit-trail is a time-sequenced record of all status changes of a business process. WSBPEL processes essentially implement a layer on top of WSDL, with WSDL defining the specific operations allowed and WSBPEL defining how the operations can be sequenced. A WSBPEL document leverages WSDL in three ways: 1) Every WSBPEL process is exposed as a web service using WSDL. The WSDL describes the entry points for external services to interact with the process, 2) WSDL data types are used within a WSBPEL process to describe the information that passes between requests, and 3) WSDL might be used to reference external services required by the process. WSBPEL processes specify stateful interactions involving the exchange of messages between partners. The state of a business process includes the messages that are exchanged as well as intermediate data used in business logic and in composing messages sent to partners. Variables (formerly called “Containers”) provide the means for holding messages that constitute the state of a business process. The messages held are often those that have been received from partners or are to be sent to partners. Variables can also hold data that are needed for holding state information related to the process and never exchanged with partners. Variables identify the specific data exchanged in a message flow, which typically maps to a WSDL message type. When a WSBPEL process receives a message, the appropriate variables are populated so that subsequent requests can access the data.

The major advantage of this approach is the seamless standard-based integration of the auditing web service with the business process. The auditing web service can be also shared among business partners and thereby enables a centrally capture of audit trail information about WSBPEL processes beyond organizational boundaries.

Although this approach enables seamless integration of the auditing services with the business process engine, there are also some trade-offs. Since the BPMSs has to invoke the auditing web service for every state change of a business process, the overhead for these web service requests can become a potential bottleneck. For instance, an order process with a large number of audits with the magnitude of thousands of process instances a day can result in potentially millions of log requests. Furthermore, this option requires a high availability of the log web service to avoid missing log requests when the process state change occurs. Nevertheless, all these issues can be solved with traditional approaches for building scalable and robust web applications [12].

McGregor and Schiefer define in [10] a method to enable business processes defined using WSBPEL to log audit information to a centralized auditing services which is part of a Solution Management Service, by establishing the Solution Management Service as a partner within the WSBPEL process definition. Nevertheless, they do not provide any details on how to automatically extend a WSBPEL process for auditing it with web services. In this paper, we want fill this gap and show an automated approach for making WSBPEL processes auditable.

4. Enabling Auditing for Existing WSBPEL

In this section, we propose a new approach to automatically enable auditing in WSBPEL processes. The basic idea is to extend a WSBPEL process definition with probe points in order to report interesting activities to an auditing web service. The resulting auditable process definition does not use proprietary elements but remains compliant with the WSBPEL standard. It is therefore possible to enhance process definitions running on different WSBPEL engines with the proposed auditing feature and centralize the collected information.

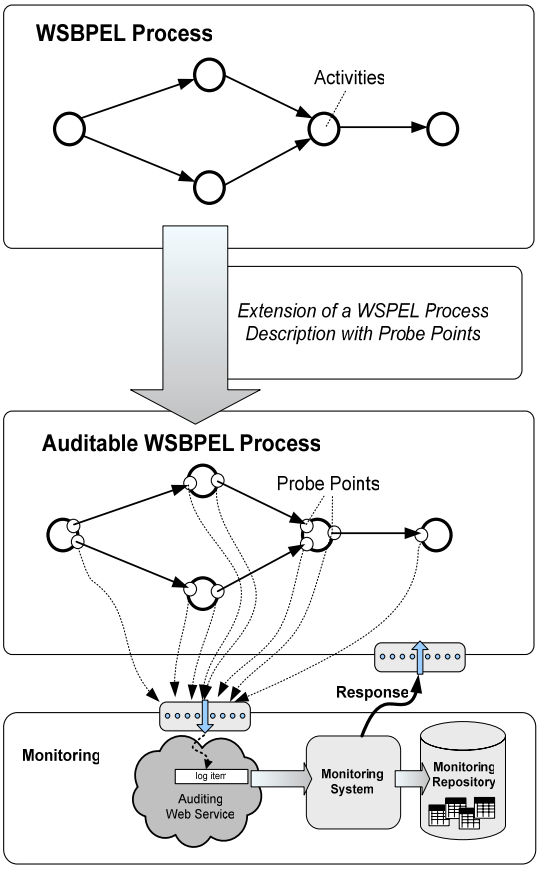

Figure 2: Creating an auditable version of a WSBPEL process

Figure 2 illustrates the proposed solution. Starting with an existing WSBPEL process we developed an XSLT stylesheet which extends the WSBPEL process with probe points. From the WSBPEL process perspective the probe points are normal activities which invoke an auditing web service. In other words, the XSLT transformation which creates an auditable version of the original process does not affect its portability.

Figure 2 shows in the middle of the figure how activities are extended, i.e. pre and post to each audited activity. Furthermore, Figure 2 shows a monitoring system which consumes and processes the audit trail data from the auditing web service. If the monitoring service discovers exceptional situations within the business process, it can respond to these situations by changing the state of the business process. When using WSBPEL, this response also can be done through a web service interface. In other words, the shown approach enables fully web-service based monitoring for business process. Please note that this paper does not discuss details on the processing of the audited information for monitoring purposes. For further details please refer to [1].

4.1. Extending WSBPEL Activities with Probe Points

We developed our auditing tool with the goal to report as much information as possible about the process state. For simplicity reasons, we focus our discussion for rest of our paper on the following WSBPEL activities:

- receive
- reply
- invoke
- throw
- pick
- switch

Prior to calling the auditing web service, the audit information needs to be assigned to the input variable which will be used to invoke the auditing web service. After the *preaudit* part, only changed contents will be updated for *postaudit*, e.g. the return variable of an *invoke* activity.

For a WSBPEL process, there may exist more than one entry point for creating a new process instance as a result after invoking a *receive* or *pick* activity with the attribute *createInstance* set to *yes*. Since these two activities represent the starting point for the instantiation of the process, no activity can be executed before them. Thus, it does not make sense to extend the process definition with a *preaudit* activity for these activities as shown in Table 1. The question mark indicates, that it depends on the activity definition whether an audit activity should be included or not. If the attribute *createInstance* is set to *yes* the preaudit step will be added otherwise it will be omitted.

A *throw* activity immediately causes the WSBPEL engine to call associated fault handlers and leaves the current branch of the process, leaving the *postaudit* useless.

Table 1: Pre- and postaudit cannot be applied to all activities

	preaudit	**postaudit**
receive	?	✓
reply	✓	✓
invoke	✓	✓
throw	✓	×
pick	?	✓
switch	✓	✓

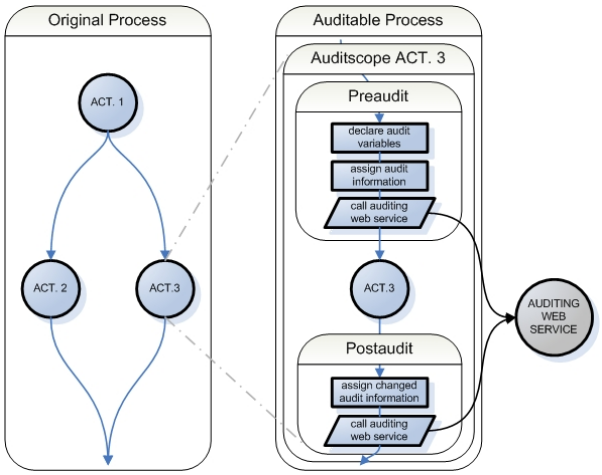

Figure 3: Auditing steps

For every audited activity, a new *scope* is created which hosts and executes all the necessary steps for *pre-* and *postaudit*, (see Figure 3). The auditing steps within this scope are as follows:

1. Variable creation
2. Preaudit variable assignment
3. Preaudit invoke statement
4. Unchanged audited activity
5. Postaudit variable assignment
6. Postaudit invoke statement

Together with the details on the examined activity, the correlation sets used during process enactment will be also transmitted as additional information to the auditing web service. The correlation information can be very important for the processing of the auditing information to recognize the process instance of the reported activities. Table 2 provides an overview which information about each of the observed activities is captured and will be sent to the auditing web service during process execution time.

```
[WebMethod]
public void auditReceive(
        string name,
        string partnerLink,
        string portType,
        string operation,
        string variableName,
        [XmlAnyElement("correlations")]
         XmlElement correlations,
        [XmlAnyElement("variableValue")]
         XmlElement variableValue)
        {
           // Processes auditing request
        }
```

Figure 4: Auditing method for receive activities

5. Auditing Web Service

The *auditing web service* is invoked whenever a *pre-* or *postaudit* part of an audited activity is reached. It provides an operation for each of the audited activities. The web service is stateless and therefore very scalable. It accepts requests from the WSBPEL process engine containing the process state information as an SOAP XML document and forwards the auditing information to subscribing target systems (e.g. a monitoring system).

For our prototype we implemented the auditing web service in C# and executed it on a Microsoft Internet Information Server. The auditing web service provides an operation for each of the audited activities. The parameters for these operations represent the message types used to declare the

Table 2: List of audited attributes

Activity	Name	partner-Link	portType	operation	(input-) Variable-Name	(input-) Variable-Value	output-Variable-Name	output-Variable-Value	correlations	*Additionally reported items*
Receive	✓	✓	✓	✓	✓	✓	x	x	✓	
Reply	✓	✓	✓	✓	✓	✓	x	x	✓	faultName
Invoke	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Pick	✓	x	x	x	x	x	x	x	✓	
Pick – onMessage	✓	✓	✓	✓	✓	✓	x	x	✓	
Pick – onAlarm	✓	x	x	x	x	x	x	x	✓	durations
Throw	✓	x	x	x	faultVar	faultVal	x	x	✓	faultName
Switch	✓	x	x	x	x	x	x	x	✓	
Switch – Case	✓	x	x	x	x	x	x	x	✓	condition
Switch – Otherwise	✓	x	x	x	x	x	x	x	✓	

variables for invoking the auditing web service. The parameters types are either *strings* or *XmlAnyElements* which can be used to store any type of XML content of a WSBPEL variable.

In the following, we show two examples for different types of web service operations which are called by the WSBPEL process engine. In Figure 4 you can see the information is passed from the WSPEL engine when auditing a *receive* activity. All data passed to this auditing web service operation represent the data that is also passed to the audited *receive* activity. In other words, any data that is available to the audited activity is also transferred to the auditing web service. In the case of the *receive* activity we capture the name of the activity, the partner link, the port type, the operation name, the variable name as well as the variable value, and the involved correlations. With this information, the auditing web service can create a complete snapshot of the data used for processing the *receive* activity.

For the second example we want to show a more complex WSBPEL activity, namely *switch* activity. Figure 5 shows how we added the probe points to the activity. As you can see in Figure 6 we used three web service operations for capturing the audit trail for various situations when processing a *switch* activity. The method *auditSwitch()* is called before the switch activity is called. It captures the name of the switch activity as well as existing correlation information. Before calling the activity in the *case* or *otherwise* branch of the *switch* activity, we insert an additional probe point for capturing the decision made by the *switch* statement. If a *case* branch of the *switch* activity is called the web service operation *auditSwitchCase()* is invoked and *auditSwitchOtherwise()* for the *otherwise* branch.

```
** PRE_AUDIT **
<switch standard-attributes>
    standard-elements
    <case condition="bool-expr">+
      ** POST_AUDIT **
      activity
    </case>
    <otherwise>?
      ** POST_AUDIT **
      activity
    </otherwise>
</switch>
```

Figure 5: Probe points for switch activities

```
[WebMethod]
public void auditSwitch(
        string name,
        [XmlAnyElement("correlations")]
        XmlElement correlations)
        {
           // Processes auditing request
        }

[WebMethod]
public void auditSwitchCase(
        string name,
        string condition,
        [XmlAnyElement("correlations")]
        XmlElement correlations)
        {
           // Processes auditing request
        }

[WebMethod]
public void auditSwitchOtherwise(
        string name,
        [XmlAnyElement("correlations")]
        XmlElement correlations)
        {
           // Processes auditing request
        }
```

Figure 6: Auditing method for switch activities

6. Extending the WSBPEL Specification for Auditing

Our proposed approach for auditing WSBPEL processes is a first step for enabling business processes with standard-based auditing mechanisms. The ultimate goal of our work is to use our experiences we gained with building auditing tools designed for WSBPEL processes for proposing effective auditing extensions for the WSBPEL process standard. An extension of the WSBPEL process standard would have the following advantages:

- By extending the schema for the WSBPEL process definition, organizations could easily specify which information of a business process should be audited.
- The web service interface for the auditing web service could be defined as part of the WSBPEL standard. This would allow tooling vendors and companies to adapt their own

implementations for handling audit trail information.

- By using a web service for the auditing, new service providers could possibly arise, who could offer services for measuring and analyzing business process performance. These service providers could offer sophisticated ways for processing process audit trails (e.g. process tracking, process mining, etc.) and also the possibility to link audit trial data of processes from multiple organizations.

6.1. Standard for Auditing Web Service

In section 0, we showed based on a simple and complex activity the operations of an auditing web service. For standardization, the WSDL has to be defined which can handle all possible requests from a WSBPEL process engine to the auditing web service. As shown in the examples in previous sections, we propose using operations for logging pre- and post *auditing* of WSBPEL activities. As we have shown for the *switch* activity, there might be the need of multiple operations for a *pre-* or *post* auditing step.

6.2. Linking the WSBPEL Process with the Auditing Web Service

If a WSBPEL process engine should be able to invoke the auditing web service, we have to define the location of the auditing service somewhere. We can do this by either defining an additional partner link in the WSBPEL process description or by introducing a new attribute or element in the WSBPEL schema. In our opinion, it would be useful to allow multiple auditing web services in a WSBPEL process definition.

6.3. Auditing Settings in WSBPEL Processes

The WSBPEL process requires additional information on which activities should be audited. Due to performance reasons, it is often desirable to only audit process activities which are relevant and of interest. For this reason, we propose two extensions for the WSBPEL standard.

The extensions include auditing settings which are included in the process description in two parts. A *centralized part* which allows to define and control the general auditing configuration, such as the location of the auditing web service(s) (in the case it is not specified as partner link), log levels (which level of details should be captured from the activities) and settings for exceptional auditing situations (e.g. what should happen if a request to the auditing web service fails).

A *decentralized part* for each WSBPEL activity which includes attributes for defining whether auditing is enabled and overwriting the default auditing settings.

```
<process ...>
    <auditing>
      <auditingServices>
        <auditingService name="Service1"
partnerLink="myAuditingServiceLink1"/>
        <auditingService name="Service2"
partnerLink="myAuditingServiceLink2"/>
      </auditingServices>
      <defaultAuditing
         audit = "true"
         auditService = "Service1"
         auditScope = "full"
         auditCorrelationSettings = "true"
         auditAsyncRequest = "true"
      />
      <faultHandler>
         <empty/>
      </faultHandler>
    <auditing>
    ...
    <invoke name="a1" ... audit="true"
      auditScope="post"/>
    ...
    <invoke name="a2" ... audit="true"
     auditScope="post"
        auditService="Service2"/>
    <invoke name="a3" ... audit="false"/>
</process>
```

Figure 7: Extension of a WSBPEL process

Figure 7 shows the proposed extensions for a WSBPEL process. The <*auditing*> element captures all centralized auditing settings and includes information about

- Available auditing web services which are linked with partner links
- The default settings for the auditing such as the default auditing web service, the auditing scope (indicating pre- or/and post auditing), the auditing correlation settings (indicating whether information about correlation sets should be captured) and auditing request settings (indicating whether the request to the auditing web service should be executed synchronously or asynchronously).

- A fault handler which will be invoked if a request to the auditing web service fails. The fault handler can include any WSBPEL activity.

Furthermore, every WSBPEL activity can overwrite the default settings defined in the <*auditing*> element. With this approach, it is possible to centrally define a default auditing behavior, but still have the possibility to define individual auditing settings for certain WSBPEL activities (e.g. skipping some activities from the auditing).

With the auditing setting in Figure 7 all activities of the process description are audited with full details on scope and correlation settings to auditing service "Service1". For the *invoke* activities with the name "a1" and "a2", only an audit tail is captured after the execution of the activities. Furthermore, instead of using the default auditing service (="Service1") for activity "a2" the auditing service "Service2" is used. For activity "a3" no audit trail information is captured. With the proposed WSBPEL extensions, companies can define auditing settings on a coarse level (e.g. process level, sequence level, scope level) as well as on a very detailed level, such as for invoke, receive, and reply activities.

7. Conclusion and Future Work

In large organizations, huge amounts of data are generated and consumed by business processes. Conventional BPMSs don't provide an open infrastructure for capturing process audit trails and the aim of this paper was to show an approach to fill this gap. This paper has presented an approach for a shareable, web service based auditing for WSBPEL processes that collects runtime information about business processes and makes them available for a broad range of applications. Additionally, we proposed extensions for the WSBPEL specification in order to include auditing mechanisms on coarse and detailed granularity levels combined with an auditing web service for logging data during process execution. The work presented in this paper is part of a larger, long-term research effort aiming at developing a service-oriented Business Process Monitoring platform.

References

[1] L. Baresi and S. Guinea, Towards Dynamic Monitoring of WS-BPEL Processes, Proceedings of the 3rd International Conference of Service-oriented Computing (ICSOC'05). Amsterdam, The Nederlands, 2005.

[2] R. M. Bruckner, J. J. Jeng, J. Schiefer, Real-time Workflow Audit Data Integration into Data Warehouse System, European Conference on Information Systems, Naples, 2003.

[3] Business Process Execution Language for Web Services version 1.1, Specification, May 2003.

[4] S. Gavirneni, R. Kapuscinski, and S. Tayur, Value of Information in Capacitated Supply Chains, Management Science, vol. 45, pp. 16-24, 1998.

[5] R. Holten, A. Dreiling, M. zur Muehlen, and J. Becker, Enabling Technologies for Supply Chain Process Management, Information Ressource Management Association International Conference, Seattle, Washington, 2002.

[6] J. J. Jeng, J. Schiefer, An Agent-based Architecture for Analyzing Business Processes of Real-Time Enterprises, proceedings of the Seventh International Enterprise Distributed Object Computing Conference (EDOC 2003), Brisbane, Australia, 2003.

[7] A. Lazovik, M. Aiello and M. Papazoglou, Associating Assertions with Business Processes and Monitoring their Execution, In Proceedings of the 2nd International Conference on Service Oriented Computing, 2004.

[8] F. Leymann, D. Roller, Production Workflow: Concepts and Techniques. Upper Saddle River (NJ): Prentice Hall, 2000.

[9] C. McGregor, S. Kumaran, An agent-based system for trading partner management in B2B e-commerce, Proceedings. Twelfth International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems, 2002.

[10] C. McGregor, J. Schiefer, A framework for analyzing and measuring business performance with web services, IEEE International Conference on E-Commerce, Newport Beach, 2003.

[11] M. zur Muehlen, Process-driven Management Information Systems - Combining Data Warehouses and Workflow Technology, presented at Proceedings of the 4th International Conference on Electronic Commerce Research (ICECR-4), Dallas (TX), 2001.

[12] D. Oppenheimer, D. A. Patterson, Architecture and Dependability of Large-Scale Internet Services. IEEE Internet Computing, pages 41–49, September/October 2002.

[13] M. Sayal, F. Casati, U. Dayal, M.-C. Shan, Business Process Cockpit, Proceedings of the 28th VLDB Conference, HP, 2002.

[14] Workflow Management Coalition (1998), Audit Data Specification.