

A Framework for Analyzing and Measuring Business Performance with Web Services

Carolyn McGregor
Centre for Advanced Systems Engineering
University of Western Sydney,
Locked Bag 1797
Penrith South DC, NSW, 1797, Australia
Email: c.mcgregor@uws.edu.au

Josef Schiefer
IBM Watson Research Center
19 Skyline Drive
Hawthorne, NY, 10606
Email: josef.schiefer@us.ibm.com

Abstract

The web services paradigm provides organizations with an environment to enhance B2B communications. The aim is to create modularized services supporting the business processes within their organization and also those external entities participating in these same business processes. Current web service frameworks do not include the functionality required for web service execution performance measurement from an organization perspective. As such, a shift to this paradigm is at the expense of the organization's performance knowledge, as this knowledge will become buried within the internal processing of the web service platform. This research introduces an approach to reclaim and improve this knowledge for the organization establishing a framework that enables the definition of web services from a performance measurement perspective, together with the logging and analysis of the enactment of web services. This framework utilizes web service concepts, DSS principles, and agent technologies, to enable feedback on the organization's performance measures through the analysis of the web services. A key benefit of this work is that the data is stored once but provides information both to the customer and the supplier of a web service, removing the need for development of internal web service performance monitoring.

1. Introduction

The emergence of e-business has dramatically changed the context in which decision-making takes place. While the fundamental human and organizational processes that take place remain largely unaffected, e-business places new constraints and demands on the decision maker to provide better service to the customers. Because of the increased rate of change possible in e-business, decisions must be made more quickly than in the past. Process

participants must have instant access to information which is relevant for the current business context. All these factors imply that the traditional decision support solutions that are focused simply on the provision of information and analysis tools are no longer sufficient. Traditional data warehouse solutions, report generators, OLAP and data mining tools typically do not allow monitoring of b2business processes on a continuous basis. To be effective, decision support must take a broader view of the whole process of decision-making that is embedded in business processes.

For a long time it was assumed that data in the data warehouse can lag at least a day if not a week or a month behind the actual operational data. That was based on the assumption that business decisions do not require up-to-date information but very rich historical data. Existing ETL (Extraction, Transformation, Loading) tools often rely on this assumption and achieve high efficiency in loading large amounts of data periodically into the data warehouse system. While this still holds true for a wide range of data warehouse applications, the new desire for monitoring information about business processes in near real-time is breaking the long-standing rule that data in a data warehouse is static except during the downtime for data loading. Workflow audit trail information is a good example for this trend because it provides the most value to the users and process analysts if it is available with minimal delays. A traditional data warehouse focuses on strategic decision support. A well-developed strategy is critical, but the ultimate value to an organization is only as good as its execution. Therefore, tactical decision support is also becoming increasingly important. A variety of architectural frameworks – such as Active Data Warehousing [20], the Corporate Information Factory [21], and Zero-Latency Enterprises [22] – have emerged recognizing the importance of tactical decision support as an extension of traditional data warehouse capabilities.

In this paper, we introduce a *web-service based framework* that aims to provide involved parties of the

business process and decision makers with comprehensive information about the status and performance of business processes independent from the type of systems that are used to execute the business process. Our approach supports a near real-time integration of audit trail data from the executed processes and thereby decreases the latency for key performance indicators and the time it takes to make the business decisions.

The remainder of this paper is organized as follows. In section 2, we discuss the contribution of this paper and related work. In section 3, we present a web-service based architecture for monitoring and analyzing business processes. Section 4 discusses in detail the web-services of the proposed architecture and introduces the concept of an Event Processing Container (EPC) which is the core component for the integration of workflow events. Finally, in section 5 we present our conclusion and discuss our future work.

2. Related work

Although monitoring and analysis are considered important tasks of the workflow management system and business process management (e.g. [15]), and the Workflow Management Coalition (WfMC) has already drafted a standard for workflow logs [16], little work has been done in developing a solution for integrating and analyzing the workflow audit trail information. Some approaches emphasize the need for integrating audit trail into data warehouse systems (e.g. the process data warehouse in [17]), others are limited to a smaller set of workflow history that is managed within a workflow management system.

Sayal et al. present in [17] a set of integrated tools that support business and IT users in managing process execution quality. These tools are able to understand and process the workflow audit trail from the HP Process Manager (HPPM), and can load via a loader component into the process data warehouse. Sayal et al. provide a high-level architecture and a data model for the process data warehouse, but they do not address the problem of integrating and analyzing the workflow audit trail in near real-time. An approach for history management of audit trail data from a distributed workflow system is also discussed in [18]. The paper describes the structure of the history objects determined according to the nature of the data and the processing needs, and the possible query processing strategies on these objects. These strategies show how to write queries for retrieving audit trail information. Unlike our approach, neither the transformation and aggregation of audit trail data, nor the analytical processing of this data are considered.

Geppert and Tombros introduce in [19] an approach for the logging and post-mortem analysis of workflow executions that uses activate database technology. The

post-mortem analysis is accomplished through querying the event history which is stored in an active database system which supports Event-Condition-Action (ECA) rules. Various types of events (e.g., database transitions, time events, and external signals) can trigger in the event history the evaluation of a condition and if the condition evaluates to true, the action is executed.

Kreger [4] defines a web service as *an* interface that describes a collection of operations that are network-accessible through standardized XML messaging. A web service provides a mechanism for organizations to carry out inter and intra organizational business processes through an XML based information exchange. Web services enable the componentization and reuse of traditional web applications. By exposing components of applications as web services and enabling businesses to invoke these components, businesses can fundamentally transform their ability to interact and engage with both current as well as potential customers and partners [1]. Web Service Definition Language (WSDL) supports the implementation of web services by providing a standard XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information [11]. Recent research into web services [1, 3] has focused on the frameworks to develop web services, but there has been little research in the area of analyzing the performance of the web service for the business performance monitoring.

McGregor and Kumaran [10] details a Solution Management framework that analyzes workflow workflow audit logs, utilizing decision support system principles and agent technologies to feedback performance measures. McGregor and Kumaran [13] builds on the work of McGregor and Kumaran [10] by applying these principles to the web services environment.

Lambros et al [6] describes the initial use of process management technology in the context of IBM web services. While this research provides significant depth in the area of publishing, locating and enacting web services, it does not provide a mechanism to enable the measurement of the performance of the web service.

3. Solution management web service architecture

Solution Management is emerging as a key component in B2B e-commerce systems. It refers to the methods and techniques used to support the monitoring of the enactment of business processes. To date, the primary role of Solution Management has been to provide visibility and control at business process level to authorized users, such as system administrators [5]. Based on our experience in working with B2B systems in real customer

engagements, we have identified the need to extend solution management capabilities to include trading partner management. If we have a network architecture in which several organizations are connected via web services, then the solution management problem is complex. In the current B2B model, solution management is contained within each organization. This is not feasible in the web services model. A request triggers other requests and so on. Traditional models of solution management fail in this model.

McGregor and Kumaran [10] define an agent-based solution manager architecture that focuses on the use of workflow enactment audit logs, utilizing decision support system (DSS) principles and agent technologies to feed back information to measure organization performance. McGregor and Kumaran [13] extend that research by transforming the interfaces to web services. They describe the principles of a Solution Management Web Service framework that provides web services to define another web service from a performance measurement perspective, and to log and analyze the enactment of that web service. A key benefit of this approach is that the data is stored once but provides information both to the organization acting as the customer and the organization acting as the supplier.

Lambros et al [6] describe the web service model by defining the interactions between the service registry, service requestor and service provider to establish and commence a web service relationship. In publishing the web service, the organization provides a service description containing the interface and implementation details including its data types, operations, binding information and network location. This definition is constructed using WSDL. Once published, potential *customers*, known also as service requestors use a *find* operation to retrieve the service description. *Customers* deciding to commence trading with a chosen supplier, found in the service registry, use the details contained in the service description to *use* that web service.

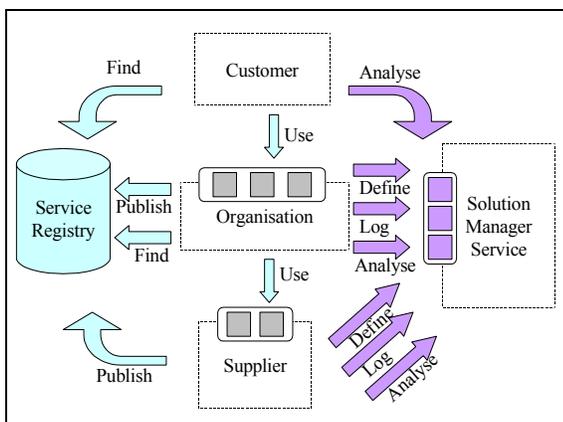


Figure 1: Solution manager web services [13]

This web services model is extended in Figure 1 to incorporate the Solution Manager Service Interface provided by the Solution Manager Service [13]. Once customers commence using web services, the Solution Manager Service provides a mechanism for any of the participants to analyze the web service performance.

In this paper we describe how process definition information can be used to execute the *define web service* to identify a web service, its owner and performance measurement information such as service level cycle times and cost to the solution manager service. This is distinct from the role of *publishing* a web service, which details the interface and implementation of the service. The solution management define web service is one of three web service categories provided by the Solution Management Web Service (SMWS) [13] architecture as shown in Figure 2.

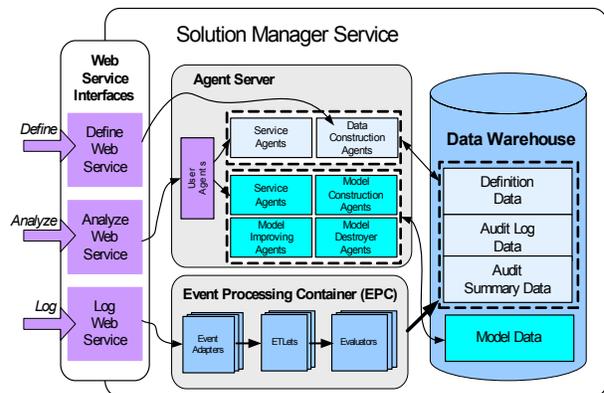


Figure 2: SMWS architecture

The Solution Management Web Services Interface provides a mechanism for organizations to define, log and analyze the web services that they participate in. The Solution Management Data Warehouse (SMDW) serves as the data repository. The Definition Data tables store data received as a result of the enactment of a *define web service*. Data received as a result of a *log web service* is stored in Audit Log Data table and, together with the definition data, is used by Data Constructor Agents to create Audit Summary Data. The Solution Management Agent Server stores the software agents that exploit the data and model data to implement solution management functions. The Event Processing Container (EPC) is responsible for integrating log data from business processes in near real-time and provides services for the parsing, translation and evaluation of workflow events. The EPC translates on-the-fly the workflow events into business process metrics which are stored in the Audit Log Data and Audit Summary Data tables. In the following section we discuss the three solution management web services in detail.

4. Solution Management Web Services

Web services within the three solution management web service type categories of define, log and analyze, are registered within the Service Registry using WSDL.

Define web service provides services to identify a web service, its owner and performance measurement information such as service level cycle times and cost. In addition, the organization may wish to establish volume targets and service prices that relate to individual customer relationships, which would also be established using one of these services. This functionality is different to that offered by the service registry as this records business level information relating to the service.

Log web service provides services to capture audit trail data from business processes. This service logs instance changes of the activity states of a workflow such as request, commencement and completion for a specific workflow instance.

Analyze web services provides a mechanism for the supplier and customer to request web service performance information. As such, two organizations have access to the same information, one as the supplier and the other, the customer.

4.1 Solution management define web service

Figure 3 provides a high level sequence diagram for the publishing and definition of a web service established by the Service Supplier to enable its customers to place orders. In publishing the web service, they provided a service description containing the details of its data types, operations, binding information and network location. This definition was constructed WSDL. In addition, this organization is required to define its web service to the Solution Management Service using WSDL.

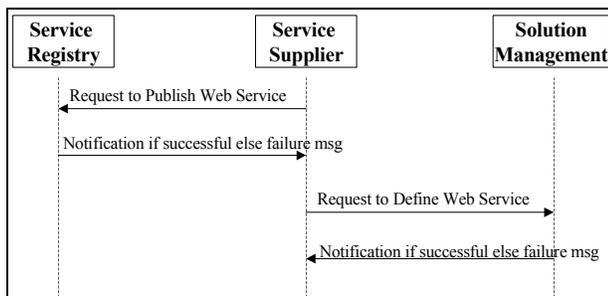


Figure 3: Web service registration and definition

The *define web service* supports two separate functions. Firstly, it provides the environment to define general performance measures for the web service. Second, it enables the definition of performance measures for a customer or customer group that will use the web service. The *define web service* details shown in Table 1

are an example of fields used when it is used to define the web service in general.

Name:	DefineWebService
Input:	<u>WebServiceDefinition</u>
	WebServiceDescription
	Web Service ID
	Description
	Owner
	OwnerID
	Name
	Address
	WebServiceStates
	StateList
	StateTransitionRules
	Product
	ProductGroup
	Product Description
	Measures
	CostDetails
	- Cost
	- StartDate
	- EndDate
	PriceDetails
	- Price
	- StartDate
	- EndDate
	CycleTime
	- CycleTimeID
	- ServiceLevel
	- ServicePercent
	- StartDate
	- EndDate
	Throughput
	- ThroughputID
	- Target
	- Frequency
	- StartDate
	- EndDate
	Rework
	- ReworkID
	- Target
	- StartDate
	- EndDate
Output:	<u>WebServiceDefinitionResponse</u>
	ResultNotification (Boolean)

Table 1: Define web service (General)

When establishing the general definition for a web service, the definition contains the name and owner of the web service, a list of the valid states for this web service and the associated transition rules for the valid state transitions to assist with the logging process. It may also contain measures relating to that web service, for example: the web service cost, price, cycle time service level and or rework targets.

Each measure has a start and end date to enable multiple values for each measure over time to be recorded where required. The *define web service* will return a flag confirming that the definition has been successful. Data constructor agents in the solution management service are triggered by the *define web service* to load this definition information into the appropriate data definition tables.

The *define web service* details shown in Table 2 are an example of fields used when the *define web service* is used to define the web service for a customer or customer group. Customer based web service definitions can be established for individual customers or groups of customers. Customer groups can be defined so as to construct customer group hierarchies and customers can belong to multiple customer groups if required.

Name:	DefineWebService
Input:	<u>WebServiceDefinition</u>
	WebServiceDescription Web Service ID
Owner	OwnerID
Product	ProductGroup
CustomerGroup	CustomerGroupID Name Description
Measures	PriceDetails - Price - StartDate - EndDate
	CycleTime - CycleTimeID - ServiceLevel - ServicePercent - StartDate - EndDate
	Throughput - ThroughputID - Target - Frequency - StartDate - EndDate
	Rework - ReworkID - Target - StartDate - EndDate
Output:	<u>WebServiceDefinitionResponse</u> ResultNotification (Boolean)

Table 2: Define web service – customer group

A customer or customer group definition for a web service contains the name and owner of the web service together with details for the customer or customer group for which this definition relates. Similar to the general definition, it may also contain measures relating to that web service, for example: the price, the cycle time service level, and or rework targets. Each of which can contain multiple values over time. Where a measure is defined for a customer or customer group, this overrides any values defined in the general definition in relation to that customer or customer group only.

4.2 Solution management log web service

The *log 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.

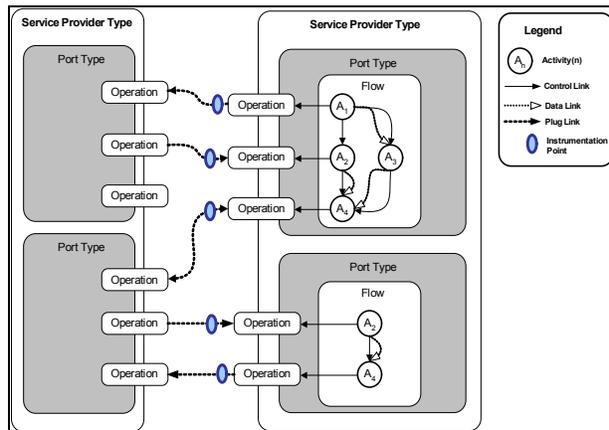


Figure 4: Instrumentation of WSFL Flows

Figure 4 shows an example with WSFL flows that include instrumentation points for web service requests. Existing web service tools can be utilized to monitor and audit web service request (e.g. IBM Web Service Gateway). A detailed discussion about the instrumentation of web services is out of the scope of this paper.

A sequence diagram for the logging process of a defined web service is shown in Figure 5. Once the definition of the web service is complete, logging can commence. Upon receipt of a request from a customer to commence a web service, a request to *log web service state* is triggered by the supplier to log the request of their web service with the Solution Management Service.

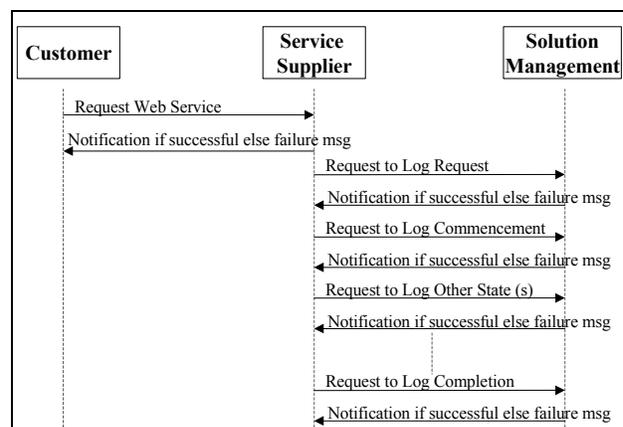


Figure 5: Web Service Logging

When the supplier commences that instance of the web service, a request to *log web service state* is triggered by the supplier to log the commencement of that instance of their web service with the Solution Management Service. As this web service instance continues towards

completion, a request to *log web service state* is triggered by the supplier to log each new status of that instance of their web service, as per the definition of the web service. When the supplier completes that instance of the web service, a request to *log web service state* is triggered by the supplier to log the completion of that instance of their web service with the Solution Management Service.

The structure of the *log web service* input, shown in Table 3 caters for multiple *WebServiceLogItems* to be contained within the one *WebServiceLogDetails* input. This enables suppliers of web services to batch their log details for performance if required.

Name:	LogWebService
Input:	<u>WebServiceLogDetails</u> WebServiceLogItem Web Service ID Customer ID Owner ID Web Service Instance ID DateTimeStamp From State ID To State ID Data of Business Objects
Output:	<u>WebServiceLogResponse</u> ResultNotification (Boolean)

Table 3: Log web service

After the log web service receives the log item information from various sources, the logging information is forwarded to the Event Processing Container (EPC). The EPC translates the raw logging data into standardized events and transforms these events into useful business information that is stored in the Audit Log Data and Audit Summary Data tables. The EPC provides a robust, scalable data staging environment for integrating audit trail data. The container approach allows a large number of events that can require complex processing logic to be handled.

Similar to Java technology for web applications, where servlets and JSPs took the place of traditional CGI scripts, our approach uses Event Adapter, ETLet, and Evaluator (see Figure 6) components that replace traditional ETL (Extraction, Transformation, Loading) solutions which very often use scripts that are hard to maintain, scale, and reuse. ETL scripts are not suitable for an event-driven environment where data extracts and data transformations are very small and frequent, because the overhead for starting the processes and combining the processing steps can dominate the execution time. Another limitation of ETL scripts is that they are written for a specific task in a self-sustaining manner, and don't provide any kind of interfaces for data inputs and outputs. Because of this constraint in the traditional approach, we use a container to manage and optimize the event processing. The EPC is a part of a Java application server and provides services for the execution and monitoring of event processing tasks. The core services are responsible for creating,

initializing, executing and destroying the managed components that are responsible for the processing of the logging data.

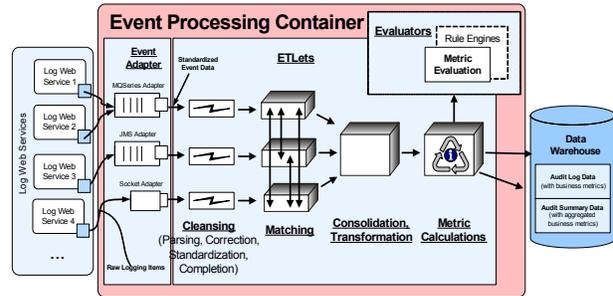


Figure 6: EPC architecture.

The EPC handles each event with a lightweight Java thread, rather than a heavyweight operating system process. Figure 7 shows the internal event processing of the EPC. The components shown with round boxes are components that are managed by the EPC. The components shown with square boxes are internal EPC components that are used to bind all managed EPC components together. Please note that the developers never see or have to deal with the internal components. We show these internal components for illustration purposes only.

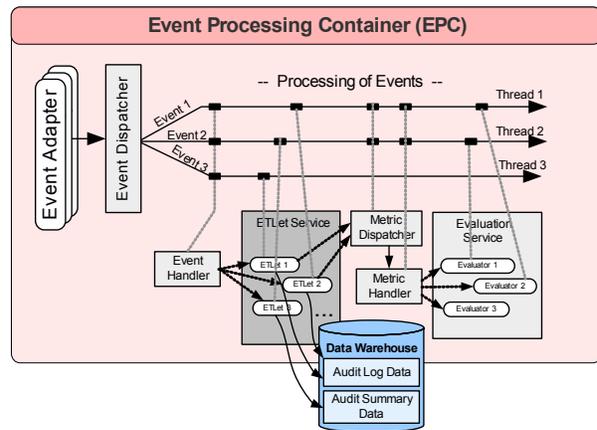


Figure 7: Multithreading within the EPC

This approach also simplifies the programming tasks for developers who have to implement the logic for the event processing, since the EPC takes responsibility for various system-level services (such as threading, resource management, transactions, caching, persistence, and so on). In our approach, we extend this concept by adding new container services, which are useful for the event processing which are not provided by other containers (e.g. EJB containers) and can be leveraged by the developers. An example of a container service is the evaluation service, which significantly reduces the effort for evaluating calculated process metrics. This

arrangement leaves the developer with a simplified development task and allows the implementation details of the system and container services to be reconfigured without changing the components. Figure 7 also shows the core components (shown as round boxes) that are managed by the EPC: 1) Event Adapters, 2) ETlets, and 3) Evaluators. The lifecycle of all components includes an *init*, *running* and *destroyed* state. Only the Event Adapters have an additional *stopped* state which enables the EPC to stop the processing of incoming events. Each of these components must implement a certain interface that is used by the EPC in order to manage the component's lifecycle. The EPC automatically instantiates these components and calls upon the interface methods during the components' lifetime. In the following subsection we discuss the managed EPC components in detail.

4.2.1 Event Adapters

The purpose of Event Adapters is to receive the raw logging items from various sources and to translate the logging items (with different logging formats) into a unified event format. Event Adapters can receive raw logging items in two ways: 1) asynchronously via messaging software, or 2) synchronously via socket connection. The first option is more scalable because it provides the ability to totally decouple the logging web-service from the actual processing of the logging data with the EPC. Every log web service must send the raw logging data to one of the Event Adapters from the EPC. A typical solution can have several Event Adapters running in parallel. They receive and dispatch events in parallel.

4.2.2 ETlets

After the dispatching of the workflow events in the Event Adapter the EPC invokes all ETlets, which are subscribed to the incoming *event type*. The subscription to event types and the configuration of ETlets can be done via a deployment descriptor. ETlets run in multiple threads in parallel. However, all processing tasks of a particular event run within the same thread. For one event type (e.g. activity events) there can be several ETlets that are executed. Note, that for the execution of all event processing steps no intermediary storage is required. The ETlet processing logic can include the calculation of process metrics and storing these metrics in the Audit Log Data or Audit Summary Data tables. ETlets can also publish the process metrics to allow the EPC to pass these metrics to Evaluator components that have subscribed to the *metric type*.

4.2.3 Evaluators

The Evaluator components can be either implemented by developers or act as proxy by forwarding the evaluation requests to rule engines for more sophisticated evaluations. In the first case, developers have to

implement the *evaluate(String metricName, Object metricValue, Map metricMetadata)* method with the evaluation logic. An Evaluator can subscribe to various *metric types* that are defined in the deployment descriptor. The EPC makes sure that an Evaluator receives the correct metrics from the ETlets. Please note, that an Evaluator isn't aware of how a metric was calculated or which events triggered the metric calculation. Therefore, the EPC enables a clean separation of extraction/receiving logic (Event Adapter), transformation logic (ETlets), and evaluation logic (Evaluators).

4.3 Solution management analyze web service

To enable performance analysis by the web service supplier and customers, the architecture supports the establishment of a series of *analyze web services*. Suppliers may access information relating to all their web service enactments, whereas customers may only access information relating to their web service enactments. Examples of questions that suppliers of a web service would like to be able to answer are:

1. Within what timeframe are we responding to tasks allocated through the initiation of a web service?
2. What web services are outside the service levels set during the definition of the web service?
3. What web services have an average response time that is degrading over time?
4. What is the volume throughput for the web service?
5. How much rework is required for the web service?
6. What is the productivity per web service?

Examples of questions by web service customers are:

1. Within what timeframe are suppliers responding to initiated web service tasks?
2. What suppliers are not responding within the service levels set during the definition of the web service?
3. What suppliers have an average response time that is degrading over time?
4. What rework is required for a web service instance?
5. How does services quality compare between similar web service suppliers, based on actual usage?

Figure 8 illustrates a sequence diagram for the request to analyze the performance of defined web services.

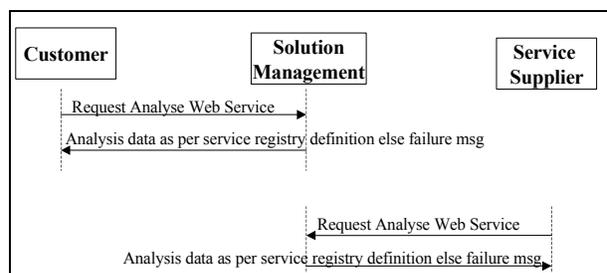


Figure 8: Web services analysis

The CycleTimeAnalysis web service shown in Table 4 requires as input the name and owner of the web service. An individual customer or customer group may also be defined. Information returned may also be restricted based on a given time interval. A key benefit of this architecture is that this same *analyze web service* can be used by both the supplier and customer of the same web service. The supplier of the web service may request information relating to all customers or select customers or customer groups. The customer however, is only able to specify themselves as the customer and may only receive information relating to their use of the web service. This access is controlled by the user's credentials controlled by the Access Control module.

Name:	CycleTimeAnalysis
Input:	<u>CycleTimeAnalysisInput</u>
	WebServiceDescription
	<i>Web Service ID</i>
	Owner
	<i>OwnerID</i>
	CustomerGroup
	<i>CustomerGroupID</i>
	Time Interval
	<i>StartDate</i>
	<i>EndDate</i>
Output:	<u>CycleTimeAnalysisResponse</u>
	OutputDataSet

Table 4: CycleTimeAnalysis web service

5. Conclusion and future work

This paper has described a framework to use process definition information to define a web service to the solution manager service. In addition, this paper describes an architecture that is capable of supporting the log web service. We introduced the concept of the Event Processing Container providing a robust, scalable and high-performance event processing environment able to handle a large number of process events in near real-time. This paper described the application of that framework within the 'Supplier Collaboration Management' use case context measuring supplier productivity and commitment. The work described in this paper can be extended in several ways for future research. Firstly, the development of an explicit link between the Workflow Management Coalition (WfMC) [7] XPD L Interface 1 [8] output and the *define web service* structure has begun. That research also includes the explicit link between the WfMC XPD L Interface 5 [9] output and the *log web service*. Second, we have commenced further analysis of the *analyze web service* categories based on the Balanced Scorecard [2] approach to organization management. Third, research has also begun to link this research with the Business Process Execution Language for Web Services (BPEL4WS) [14].

6. References

- [1] Bosworth, A., Developing Web Services, Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, 2001.
- [2] Kaplan R.S. and Norton D.P., The Balanced Scorecard - Measures That Drive Performance, Harvard Business Review, Jan-Feb, pp 71-79, 1992.
- [3] IBM, Web Services Development Concepts, IBM Software Group, IBM T.J. Watson Center, Yorktown, NY, 2001.
- [4] Kreger, H., Web Services Conceptual Architecture (WSCA 1.0), IBM Software Group, 2001.
- [5] Kumaran, S., Bhaskaran, K., Chung, J., Das, R., Heath, T., Nandi, P., An e-Business Integration & Collaboration Platform for B2B e-Commerce, 3rd International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems, IEEE Computer Society Press, 2001.
- [6] Lambros, P., Schmidt, M., Zentner, C., Combine Business Process Management Technology and Business Services to Implement Complex Web Services, 2001.
- [7] Workflow Management Coalition Workflow Reference Model, Document Number WfMC-TC-1003, 1995.
- [8] Workflow Management Coalition Workflow Process Definition Interface - XML Process Definition Language, Document Number WfMC-TC-1025, 2002.
- [9] Workflow Management Coalition Audit Data Specification, Document Number WfMC-TC-1015, 1998.
- [10] McGregor, C. and Kumaran, S., An Agent-Based System for Trading Partner Management in B2B Commerce, Proc. of the 12th International Workshop on Research Issues on Data Engineering (RIDE-2EC'2002), San Jose, USA, 2002.
- [11] W3C, Web Services Description Language (WSDL) 1.1, 2001.
- [12] IBM, Business Process Execution Language for Web Services, Version 1.0, 2002.
- [13] McGregor, C. and Kumaran S., Business Process Monitoring Using Web Services in B2B Commerce, IEEE 2002, Proc. 2nd International Workshop on Internet Computing and E-Commerce (ICEC'02), Fort Lauderdale, USA, 2002.
- [14] Business Process Execution Language for Web Services, Version 1.0, 2002.
- [15] Jablonski, S., Bussler, C., Workflow Management. Modeling Concepts, Architecture, and Implementation. Intl. Thomson Computer Press, London, 1996.
- [16] Workflow Management Coalition Audit Data Specification, Document Number WfMC-TC-1015, 1998.
- [17] Sayal, M., Casati, F., Dayal, U., Shan M., Business Process Cockpit, VLDB 2002, Peking, 2002.
- [18] Koksai, P., Alpinar, S. N., Dogac, A. Workflow History Management, ACM Sigmod Record 27(1): 67-75, 1998.
- [19] Geppert, A. and Tombros, D., Logging and Post-Mortem Analysis of Workflow Executions based on Event Histories. Proc. 3rd Intl. Conf. on Rules in Database Systems (RIDS), LNCS 1312, Springer Verlag, Heidelberg, Germany, pp 67-82, 1997.
- [20] Brobst, S. A., Ballinger, C., Active Data Warehousing, Whitepaper EB-1327, NCR, 2000.
- [21] Inmon, W. H., Imhoff, C., Sousa, R., Corporate Information Factory, Second Edition, J. Wiley and Sons, New York, 2001.
- [22] Gartner Group, Introducing the Zero-Latency Enterprise, Research Note COM-04-3770, 1998.