Conceptual Framework for a Comprehensive Service Management Middleware

Farhana Zulkernine and Patrick Martin
School of Computing, Queen's University, Kingston, ON, Canada
{farhana, martin}@cs.queensu.ca

Abstract

Web services have greatly leveraged the world of Business-to-Business (B2B) communication and promise a lot more through dynamic service composition. In order to compose Web services into a business process, it is necessary to find suitable services, negotiate and establish Service Level Agreements (SLAs) with the service providers, build a workflow with the selected services and finally execute the workflow while monitoring the performance of the services to verify that the SLAs are satisfied. In this paper, we propose a conceptual framework for a Comprehensive Service Management Middleware (CSMM) to leverage the usability of Web services in business processes. CSMM has a distributed and modular architecture suitable for the distributed nature of Web services and Service Oriented Architecture (SOA) in general. The novel approach of architecting the middleware based on Web service technology can leverage outsourcing of management responsibilities, and thereby, make the most cost effective use of Web services.

1. Introduction

Web services are independent software that are accessible through standard interfaces and provide specific services over the Internet. There are several steps for using a Web service in a business process which can add considerable overhead depending on the type and usage of services, and the complexity of the process. The following steps, as depicted in Figure 1, are common for all service consumers; however, the complexity of each step may vary.

- Service Selection: Select a service based on some predefined criteria to complete a business process or replace a service to recover from failure.
- SLA Negotiation: Negotiate the SLAs based on customer requirements and service offerings.
- Workflow Orchestration: Design a workflow for a business process by organizing selected Web services with properly matched input and output parameters.
- Workflow Execution: Ensure possible error check points, alternative paths to handle exceptions, and corrective measures in the workflow, and thereby, execute it.
- Monitor and Error Report: Monitor the services' performance to verify compliance with the SLAs and optionally report QoS information to a specified knowledgebase to enable quality-based service selection. Also check for possible failure to allow quick recovery.

All of the tasks listed above present non-trivial problems for composite business processes and can incur significant overhead for both the service providers and the service consumers. Ideally a consumer should be able to use a service like a function call and the additional tasks can be executed by a management service provider in order to simplify and facilitate the use of Web services. In the absence of a management service provider, the service consumers would have to write and maintain proprietary code for the above operations. In this paper, we propose a conceptual framework called the *Comprehensive Service Management Middleware (CSMM),* where separate modules can provide client-side management services at different steps, thus providing one comprehensive management service to the service consumer.

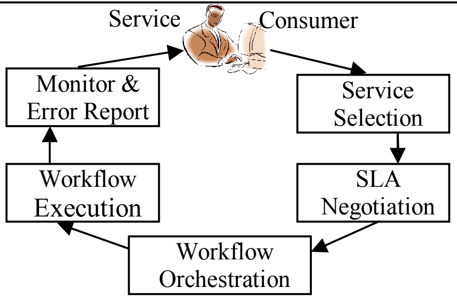

Figure 1. Steps to execute a Web service process

The rest of the paper is organized as follows. Section 2 illustrates the problems involved in each of the five steps outlined above, namely selecting,

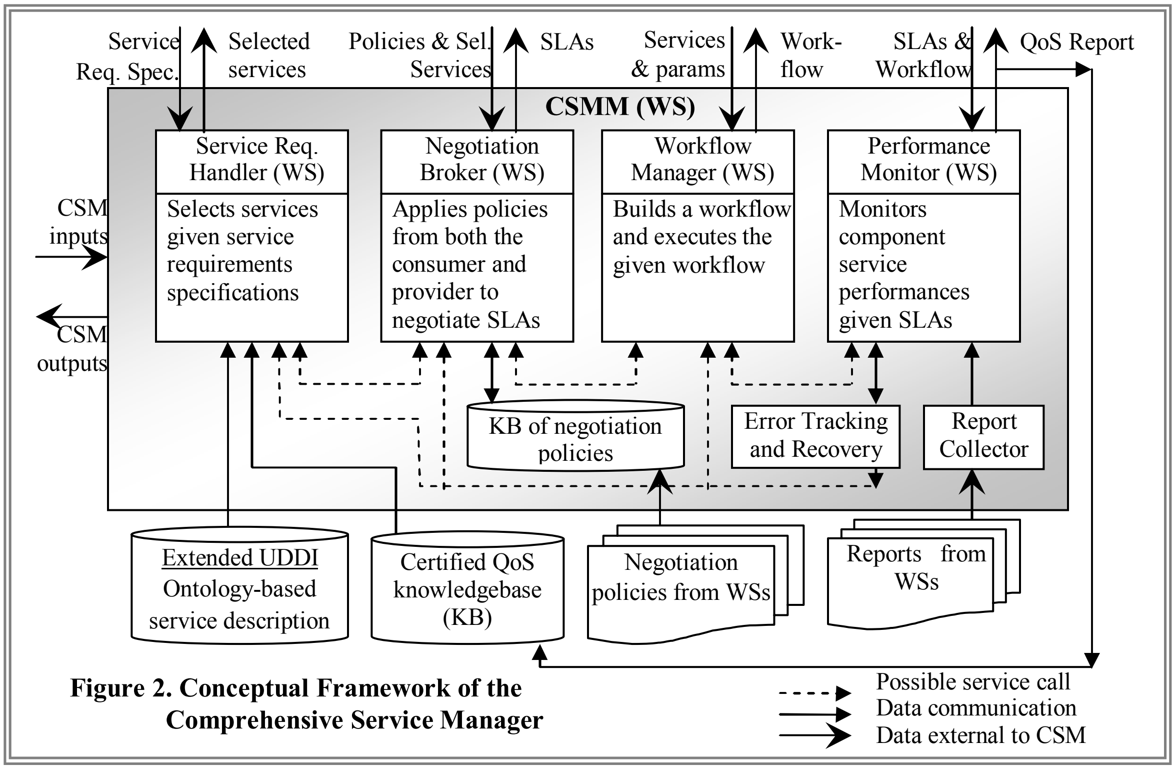
Service Req. Spec.
Selected services
Policies & Sel. Services
SLAs
Services & params
Work-flow
SLAs & Workflow
QoS Report
CSMM (WS)
Service Req. Handler (WS)
Selects services given service requirements specifications
Negotiation Broker (WS)
Applies policies from both the consumer and provider to negotiate SLAs
Workflow Manager (WS)
Builds a workflow and executes the given workflow
Performance Monitor (WS)
Monitors component service performances given SLAs
CSM inputs
CSM outputs
KB of negotiation policies
Error Tracking and Recovery
Report Collector
Extended UDDI
Ontology-based service description
Certified QoS knowledgebase (KB)
Negotiation policies from WSs
Reports from WSs
Possible service call
Data communication
Data external to CSM

Figure 2. Conceptual Framework of the Comprehensive Service Manager

negotiating, orchestrating, executing, and monitoring services to substantiate the necessity of a comprehensive service manager. Section 3 gives an overview of the CSMM. How the framework can be used for a composite Web service process is explained by a scenario in Section 4. Section 5 discusses the existing solutions and the related work on service management. A summary and discussion of the future work concludes the paper in Section 6.

2. Problems in Client-side Service Management

With the growing popularity of Web services, the complexity in each step for creating and executing a Web service process has increased considerably. A service, as described in the requirement specification, may need to be divided into smaller tasks so that matching services can be found. Data conversions and formatting may be necessary to link outputs from one service to the inputs of another service. Service selection criteria can include functional properties, such as performance, reputation, context, and parameters. An ontology specification [12] is deemed inevitable to enable automatic matching of service selection criteria with the service properties. Context matching [8] [11] in service selection is gaining popularity in the area of ubiquitous and pervasive computing.

Negotiation of the Service Level Agreements (SLAs) is important for business processes, particularly when there is a service charge. Negotiation is done based on the service offerings, both functional and non-functional, such as price, quantity, date, availability of the service, throughput, response time, delay, and may include bonus offers. Usually, bilateral bargain type negotiations [4] [9] are performed for Web services, which can consume considerable bandwidth and processing time. The priorities for trade-offs between different issues of negotiation and the service offerings may vary depending on the context of the negotiating parties.

Some of the challenges in building a workflow are verifying the types and compatibility of the parameters of the adjacent Web services in the workflow and linking them properly, establishing check-points at positions in the workflow where errors can most likely occur, and setting the recovery path to possibly continuing the workflow.

Usually the performance of a Web service is monitored to ensure SLA compliance at both the service provider's end [3] and at the consumer's end. However, service consumers face a greater challenge in monitoring composite services due to the chaining of the function calls and responses, and the network dependency in measuring some of the functional attributes such as reliability, response time, and service delay [6].

Outsourcing of the above management tasks can reduce the overhead experienced by the consumers and leverage the use of Web services. Although multiple partial solutions have been proposed, for coherent and seamless management of these tasks a coordinated management framework such as CSMM, is essential.

3. Overview of the Framework

We propose the CSMM as a solution to the problems of client-side management. It contains several distributed modules and knowledge repositories as shown in Figure 2, which together provide complete management functionality as a Web service. Implementation of each of the four main modules in CSMM presents a significant research problem in this area and will be further expanded as the research progresses. In this paper, we present our ideas about how each of these modules can be implemented. The four main modules in the framework are designed as independent Web services and can be invoked separately. We plan to implement these Web service modules based on the Autonomic Web Service Environment (AWSE) [18] framework, which provides autonomic control of the Web service at the provider's end. We describe the modules below in further detail.

3.1 Service Requirements Handler (SRH)

Service Requirements Handler finds required services for the user based on some specified selection criteria. It accepts specifications describing service requirements in a formal language and returns a set of selected services in the order of execution. The language should be specified based on the XML (eXtensible Markup Language) [21] and Web service ontology, which we will refer to as a Service Requirements Specification Language (SRSL). The Semantic Web Services group is currently working on Web Ontology Language for Services (OWL-S) [15], which supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of Web services in an unambiguous and computer-interpretable form. For service discovery, semantic markup information about service offerings, properties, parameters, and return values, should be stored in an extended UDDI. A SRSL based on standards like OWL-S, can match the semantic service selection criteria against the information in the extended UDDI. To enable QoS-based service selection, we propose the use of a certified QoS knowledgebase that can be built by the Performance Monitor in CSMM as shown in Figure 2. In case of an error in a workflow, SRH can find a replacement service. For complete automation, SRH communicates directly with the Negotiation Broker.

```
<SOAP-ENV:Body>
   <reportLog xmlns = 'urn:ws:reportLogs'>
      <receiver>MonitorService</receiver>
   </reportLog>
</SOAP-ENV:Body>
```

Figure 3. SOAP message sent to a service

```
<SOAP-ENV:Body>
   <reportLog xmlns = 'urn:ws:reportLogs'>
      <receiver>MonitorService</receiver>
      <service>MyLocationServices</service>
      <repTime>1029200613:11:06</repTime>
      <report>success</report>
      <retval>Niagara Falls</retval>
   </reportLog>
</SOAP-ENV:Body>
```

Figure 4. SOAP message with reports

3.2 Negotiation Broker (NB)

Negotiation Broker takes an ordered list of selected services and the negotiation policies from all the service providers and the service consumers. The policies specify the context of the negotiators, their goals, constraints, preferred strategies and external factors that may influence the decision process as shown in the example in Figure 6. A knowledgebase stores the negotiation policies, which can be used to derive improved negotiation strategies and provide assistance in the case of uncertainties in negotiation issues using artificial intelligence techniques. Also stored policies can be retrieved for subsequent negotiations between the same consumer-provider pair. This module performs the negotiation locally as a broker service and returns SLAs to both parties. This can reduce network traffic, and security issues in negotiation. However, the negotiating parties have to trust the broker to convey their goals and policies. We assume that the NB is a trusted service and WS-Trust [22] can be used as a guideline for the trust relationship. The module can be expanded to multiple sub-modules for strategic decision making, multiple SLA negotiation for composite processes, and SLA generation. NB can be extended to perform more general negotiations.

3.3 Workflow Manager (WM)

Workflow Manager takes an ordered list of selected services with the necessary input parameters for each of them, and generates a Web Service Business Process Execution Language (WS-BPEL) [13] specification of a service orchestration. The workflow can be returned to the customer to be executed locally or can be executed by the WM in a manner similar to the workflow execution engine in BPEL. Orchestration describes how Web services can interact with each other at the message level, including the business logic and execution order of the interactions. These interactions may span applications and/or organizations, and result in a persistent, transactional, multi-step process model. WM is designed as a Web service. WM defines the workflow with check-points for exception handling and monitoring purposes and executes it in a centralized manner as the BPEL workflow engine, which allows it to handle the exceptions or re-invoke services. The output from the last Web service in a process can go directly to the customer. To achieve higher reliability, WM can re-invoke a service with fewer constraints if it does not return a result in the first place, or design the workflow to generate a set of alternative results for the customer.

3.4 Performance Monitor (PM)

Performance Monitor takes the SLAs and workflow specification as input and performs the SLA compliance checking for the service consumer. If a service becomes unavailable, it should react after some threshold period and request the Error Tracking and Recovery (ETR) sub-module to take a corrective action based on the workflow definition. For example, the ETR can request SRH for a service replacement and then request WM to build a revised workflow with the replaced service. A Report Collector module collects performance reports from services for the PM, which in turn analyses the reports and sends the analyzed data to the certified statistical QoS knowledgebase to enable QoS-based service selection.

Our design requires each individual service of a composite workflow to report back to the monitor. This could be achieved by the extension of the Simple Object Access Protocol (SOAP) [20] message structure to include a "reportLog" as shown in Figures 3 and 4. The interpreter of SOAP messages at the service provider's end will extract this log part from the message, which specifies where the log report should be sent. The module invoking a service would initialize the "receiver" attribute within the "reportLog" to specify the party that would receive the report. The SOAP message interpreter would be responsible for returning a SOAP message to the designated receiver and specifying the other fields within the "reportLog" such as, "service" referring to the service name, "repTime" referring to the time when the service completed or the reporting time, "report" notifying "success" or "failure", and possibly the return values for the "retval" attribute at the end of the service execution.

The report can contain minimal information to limit the network traffic and help trace the process workflow. It may be sufficient to store the last report only to reduce storage of state information. This is, however, a stateful and centralized approach. In a more stateless and decentralized approach, the logs would continue to be added to the SOAP message, which would be passed along the process, and the final Web service would send it to the originator of the process. In the latter approach, the risk lies in the report being lost with the failure of a service, resulting in a longer timeout period for the originator to detect a failure in the workflow.

4. Example Scenario of CSMM

We describe the CSMM with an example scenario of a customer wanting to plan a vacation using Web services. This will typically require multiple services such as site selection, travel planning, hotel reservation, and tourist services, to be chained together in a composite service process. CSMM can assist the customer in creating, executing and managing this composite service. Since service requirements specification is the key that guides the activity of CSMM, it should be specified properly. An intelligent user interface with a knowledgebase of possible services and service options can facilitate the specification. SRH finds component services to meet the specifications and returns the services in the order of execution either to the customer, if SRH service was invoked, or to the NB, if the comprehensive service of CSMM was invoked. Figure 5 shows an example of the type of information that the service consumer should convey to the SRH through a SRSL specification.

The NB then collects the policies from both parties for each service, and places them into the negotiation knowledgebase. Figure 6 shows an example of a negotiation policy specification of the "selectLocation" service provider. It contains the name of the policy owner as "policymaker", the "context" of the party which is considered by the other party during negotiation, "goal", "constraints" and negotiation "strategy". The constraint specification may include external factors defined as functions which should be evaluated by the NB during the negotiation process, such as the currently available resources. Other

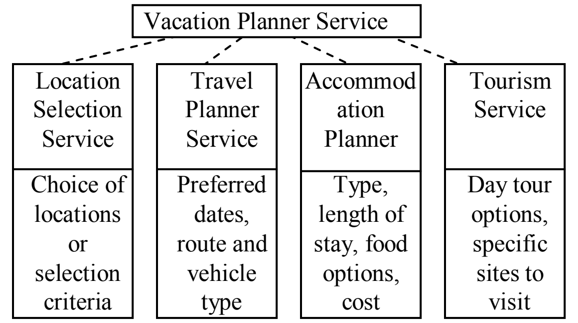
Vacation Planner Service
Location Selection Service
Travel Planner Service
Accommodation Planner
Tourism Service
Choice of locations or selection criteria
Preferred dates, route and vehicle type
Type, length of stay, food options, cost
Day tour options, specific sites to visit

Figure 5. Service requirement specification basics

```
<policyMaker>selectLocation</policyMaker>
<context>new service</context>
<goal>get more service contracts</goal>
<constraints>
  <externalFactors>getResourceAvailability()<90%
   </externalFactors>
   <maxAvailability> 99.5%</maxAvailability>
   <minPrice> $0.5/hr/connection</minPrice>
</constraints>
<strategy>
   <condition>
      <if>number of connections <100 or
           servicePeriod>30days or
           customerContext = small customer </if>
      <then><price> $0.5/hr/connection </price>
              <reliability>99%</reliability> </then>
      <else><price> $0.8/hr/connection</price>
              <reliability>98%</reliability> </else>
   </condition>
   <tradeoff><incAvailability>=90%> 1%
         <incPrice>$0.2/hr/connection</incPrice>
      </incAvailability>
   </tradeoff>
</strategy>
```

Figure 6. Negotiation policy specification for the "selectLocation" service provider

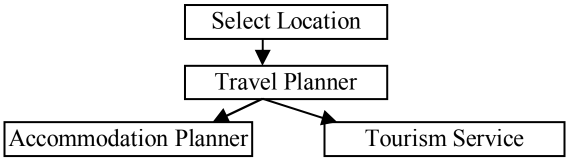

Figure 7. Workflow for vacation planning service

constraints may specify maximum and minimum values of different service attributes. A negotiation strategy may describe conditional prices based on the service demand and required service attributes. The "tradeoff" section specifies the incremental rate of the service price beyond a certain point of desired service property. In the above example, when the service availability is >= 90%, a $0.2 price increment incurs for every 1% increase in service availability. If a negotiation completes successfully then a set of SLAs are sent back to both the parties involved. For our example, negotiation is done for all four services and if a negotiation fails, no SLAs are returned for it. When a comprehensive service is expected, the NB requests the SRH for a replacement service.

Figure 7 shows a very simple and straight forward workflow of our example vacation planning composite service process. The arrows indicate information and control flow and the square boxes show each service in the process. The WM then builds a workflow with the selected services in sequence, and parameters that are obtained from the customer such as different options and preferences. Some of the parameters are defined by the outputs of the previous services in the workflow. Output from the "selectLocation" service is passed as the input to all other services. Once the location is selected, that information along with other user preferences, are sent to the travel service to book travel media. As the travel plans are made, the dates and location information are used for the other two services to arrange for accommodation and plan tours. Since the latter two are independent of each other, they can be executed in parallel.

The "reportLog" section (ref. Figure 3 and 4) is inserted in the message sent to the location selection service for monitoring purposes. By default, the PM is designated as the receiver of the reports. However, the customer can also choose to receive the reports. The PM should obtain a message from each service in the workflow as the services return the results. The reports should contain necessary information to verify compliance with the SLAs. A missing report indicates a failure or unavailability on the service's part and necessary corrective actions are initiated by the ETR sub-module. In case of a successful execution, the location selection service would select a location, the travel planner service would book or buy tickets for traveling to that location, the accommodation planner service would book the hotel and finally, the tourism service would book tours for the customer.

5. Related Work

To the best of our knowledge, no other work in the literature addresses the complete client-side service management problem. Web Service Management Layer (WSML) [1] is a middleware that lies between the client application and Web services, and facilitates development and client-side management of integrated service applications. The modules in CSMM are designed as Web services and have a more distributed architecture. Illner *et al.* [5] present their work on policy governed automated management of embedded service systems using model-based approach. Our work uses policies only for the NM, and focuses on managing and executing consumer calls rather than managing Web service systems. Universal Service Description Language (USDL) [17] and Web Service Offering Language (WSOL) [19] are some of the work that are geared towards more efficient service selection and management and may contribute to the definition of a SRSL. Other comparable work in this area includes specification of QoS ontology for autonomic service selection using agents as proxies [12], study of the requirements for context representation for Web services [11], and processing heterogeneous context information [8]. Gimpel *et al.* [4], Comuzzi *et al.* [2], and Li *et al.* [9] propose different Web-based negotiation approaches that are comparable to the approach of the NB module in CSMM. Sahai *et al.* [16] propose a distributed message tracking algorithm and the Web Services Management Network (WSMN) [9] for monitoring composite Web service processes. The message tracking used in the CSMM PM differs from that in the formatting and processing of the messages. Other research work that contributes to client-side management to some extent includes QoS-based service composition [23].

6. Conclusion

CSMM can relieve the customers from the overhead of performing the management tasks for building and executing composite Web services based workflows. The conceptual architecture of the CSMM contains four main modules for Web service selection, workflow definition and execution, SLA negotiation, and client-side monitoring. The modules are designed as Web services, which allow individual or comprehensive use of their services as required by the consumer, and also extend the usability of the modules to provide other similar services.

We are working on designing and implementing CSMM, starting with the negotiation module, for use in our AWSE framework [18]. As future work, CSMM modules can be further extended for use in the paradigm of wireless or pervasive computing and small industrial devices [7], where clients have limited processing power. Due to the use of Web service technology, CSMM can also provide ubiquitous access to a wide range of service consumers. The individual service modules in CSMM can be further utilized as independent services for automated policy based negotiations, service ratings, and distributed resource monitoring with WSDM (Web Services Distributed Management) interfaces [14]. Thus CSMM can leverage the use of Web services in business processes and provide modular services for similar tasks to a wide range of customers.

References

[1] Cibrán, M., Verheecke, B., Suvee, D., Vanderperren, W., and Jonckers V. (2004). "Automatic Service Discovery and Integration using Semantic Descriptions in the Web Services Management Layer", *Journal of Mathematical modeling in Physics, Engineering and Cognitive Sciences*, v. 11, p.79-89.

[2] Comuzzi, M., and Pernici, B. (2005). "An Architecture for Flexible Web Service QoS Negotiation", in *Proc. of the 9th IEEE Int. EDOC*, Enschede, The Netherlands, p.70-82.

[3] Dan, Davis, Kearney, Keller, King, Kuebler, Ludwig, Polan, Spreitzer, and Youssef (2004). "Web services on demand: WSLA-driven automated management", *IBM Systems Journal*, v. 43(1), p.136-158.

[4] Gimpel, H., Ludwig, H., Dan, A., and Kearney, B. (2003). "PANDA: Specifying Policies for Automated Negotiations of Service Contracts". Orlowska, M., Weerawarana, S., Papazoglou, M., and Yang, J. (Eds.): *LNCS 2910,* p. 287–302, Springer-Verlag Berlin Heidelberg.

[5] Illner, S. Pohl, A. Krumm, H. Luck, I. Manka, D. Stewing, F.-J. (2006). "Policy-based self-management of industrial service systems". In *Proc. of the 4th Int. Conf. on INDIN,* Singapore, p. 492-497.

[6] Iyengar, A., King, R., Ludwig, H., and Rouvellou, I. (2003). "Performance and Service Level Considerations for Distributed Web Applications", in *Proc. of the 7th World Multi-conference SCI*, Orlando, Florida.

[7] Jammes, F. and Smit, H. (2005). "Service-Oriented Architectures for Devices- the SIRENA View", in *Proc. of the 3rd IEEE Int. Conf. on INDIN*, p.140-147, Perth, Australia.

[8] Kranenburg, H. van, and Eertink, H. (2005). "Processing Heterogeneous Context Information", in *Proc. of the IEEE SAINT-Workshop,* p.140-143.

[9] Li, H., Su, S.Y.W., and Lam, H. (2006). "On Automated e-Busines Negotiations: Goal, Policy, Strategy, and Plans of Decision and Action." *Journal of Organizational Computing and Electronic Commerce*, v. 13(1), p.1-29.

[10] Machiraju, V., Sahai, A., and van Moorsel, A. (2002). "Web Services Management Network: An Overlay Network for Federated Service Management", HP Technical Report HPL-2002-234.

[11] Martin, D. (2006). "Putting Web Services in Context", in *Elec. Notes in Theoretical Comp. Science*, v. 146(1), p.3-16.

[12] Maximilien, E. M., and Singh, M. P. (2004). "A Framework and Ontology for Dynamic Web Services Selection", *IEEE Internet Computing*, v. 8(5), p.84-93.

[13] OASIS WS-BPEL (2006). Available: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel.

[14] OASIS WSDM and MUWS v.1.1 *(*2006). Available at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm.

[15] OWL-S 1.2 2006. (http://www.daml.org/services/owl-s/.)

[16] Sahai, A., Machiraju, V., Ouyang, J., and Wurster, K. (2001). "Message Tracking in SOAP-based Web Services", Hewlett Packard. Tech. Rep. HPL-2001-199.

[17] Simon, L., Mallya, A., Bansal, A., Gupta, G., Hite, T. D. (2005). "A Universal Service Description Language" in *Proc. of the IEEE ICWS*, Orlando, Florida, USA.

[18] Tian, W., Zulkernine, F., Zebedee, J., Powley, W., and Martin, P. (2005). "An Architecture for an Autonomic Web Services Environment", in *Proc. of WSMDEIS*, Miami, Fl.

[19] Tosic, V., Pagurek, B., Patel, K., Esfandiari, B., and Ma, W. (2005). "Management applications of the Web Service Offerings Language", *Information Systems,* v.30(7),p564-586.

[20] W3C SOAP 1.2 (2004). Part 1: Messaging Framework. Available at: http://www.w3.org/TR/soap12-part1/.

[21] W3C XML (eXtensible Markup Language). Available at: http://www.w3.org/XML/.

[22] WS-Trust, 1.0 (2002). Available at: http://www.verisign.com/wss/WS-Trust.pdf.

[23] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q. (2003). "Quality driven Web services composition", in *Proc. of Int. WWW,* Budapest, Hungary.