A QoS Oriented Framework for Adaptive Management of Web Service based Workflows

Chintan Patel, Kaustubh Supekar, and Yugyung Lee

School of Interdisciplinary Computing and Engineering
University of Missouri-Kansas City
{copdk4, kss2r6, leeyu}@umkc.edu

Abstract. Web Services are emerging technologies that enable application-to-application communication and reuse of autonomous services over Web. Traditional Workflow Management Systems fail to provide a comprehensive solution for a Web Service based Workflow. A framework that meets the quality of service (QoS) requirements for ad hoc Internet based Services is rarely provided. Considering the increasing demand for expanding services and application requirements coupled with use of Web Services, it is a challenging task to develop a QoS model as a framework for Web Service based Workflows. In this paper, we have proposed a QoS oriented Framework, called WebQ, that is capable of conducting the adaptive selection process and simultaneously provides binding and execution of Web Services for the underlying workflow. To achieve these objectives, as the first step, we have designed a QoS model for Web Service selection, binding, and execution. We, then, develop a set of algorithms to compute QoS parameters and implement them using a rule-based system. A series of experiments performed on workflows composed of real Web Services have confirmed that the proposed framework is very effective in improving the overall QoS of the system.

1 Introduction

Web Services have been used to achieve system interoperability over the Web through exchange of application development and service interactions using the standards like Web Services Description Language (WSDL), Universal Description, Discovery and Integration (UDDI) [1], and Simple Object Access Protocol (SOAP) [2].

While current Web Service technologies show much progress, the current services are mainly limited to atomic services [3]. Thus, it is not adequate to handle the autonomous and complex services in realistic settings. In dealing with this problem, some research work [4] has developed languages to compose the individual Web Services into transactions or workflows. Web Services Flow Language (WSFL) was designed for service compositions in the form of a workflow [5], XLANG [6] for the behavior of a single Web Service. However these works are not sufficient for providing the adaptive web Services generated from a particular context.

In fact, the automatic or semi-automatic management of service flows over the Web has not been achieved. For the purpose, we need to examine the characteristics of the composition model of Web Services. In the Web Services model that is quite different from traditional one, there is the large number of similar or equivalent services which user can freely select and use for their application. Second, since the service is developed and deployed by the third party, the quality of service is not guaranteed. Third, the services may not be adequate as per service requestor's requirements and kept evolving, without notification to service requestors, according to the provider's requirements and computing environment. Thus, it is important to provide adaptability to evolving services as well as diverse context of services.

Kammer et al. [7] suggested workflow to be dynamic, which allows changes with minimal impact to the ongoing execution of underlying workflow, as well as be reflexive, which provides knowledge about a workflow's applicability to the context and the effectiveness of its deployment evaluated over time. Understanding constraints and context associated with services may affect the quality of service. From this perspective, optimization may occur through the evaluation and refinement of a previous service flow. Facilitating service composition may not be difficult if one knows which Web services and in which order to compose. However, automatic composition of services is challenging. It is because it is difficult to capture semantics and context of services and measure the quality of services. One exemplary effort that aims for this function is DAML-based Web Service Ontology (DAML-S) [8], describing the properties and capabilities of Web services [9].

The goal of this project is to develop a QoS-based web service framework, called WebQ, that enables to select appropriate Web services, dynamically bind the services with the underlying workflow, and perform the refinement of existing services. Since evaluation and refinement of services tend to be subjective, the generation of them needs a formal matrix, which can measure the requirements and quality of services. In the WebQ Framework, criteria is quantitatively set to evaluate whether a particular service is appropriate for specific service flow and to define service quality. This generic framework can be applicable to the various domains, such as e-commerce, medicine, and bioinformatics.

2 Related Work

Workflow technology has been around since a decade and has been successful in automating many complex business processes. A significant amount of work has been done in this field which deal with different aspects of workflow technology viz process modeling, dynamic workflows [10], distributed workflows [11]. Process modeling languages such as IDEF, PIF [12], PSL [13] or CIMOSA [14] and frame based models of services were used to design process typing, resource dependencies, ports, task decomposition and exception.

On the other hand Web Services are emerging resources on the web and have received a great deal of attention from the industry. Standards such as WSDL,

SOAP, and UDDI are being developed for low-level descriptions of Web services. Web Services Description Language (WSDL) provides a communication level description of the messages and protocols of services [15]. Simple Object Access Protocol (SOAP) invokes services through remote method invocations over HTTP [2]. Universal Description, Discovery and Integration (UDDI) [1] announces and discovers services which are registered by their providers at a logically central registry. Some emerging approaches include additional features through the use of ontologies, e.g.describing and indexing services based on process models [16].

Current research on web services paves way for web service based workflows, which has obvious advantages pertaining to scalability, heterogeneity, reuse and maintenance of services. Major issues in such inter-organizational service based workflows are service discovery, service contracts [17] and service composition. Web Services Flow Language (WSFL) was proposed to describe compositions of services in the form of a workflow, which describes the order of service invocation [5]. Service composition aids such as BizTalk [18] were proposed to overcome the limitations of traditional workflow tools [14] which manually specify the composition of programs to perform some task. Other industrial initiatives such as BPEL4WS [19], XLANG [6] concentrate on service representation issues to tackle the problem of service contracts, compositions and agreements. Current efforts [20] are to automate the complete process of service discovery, composition and binding using machine understandable languages. Some other recent advances are WS-Transaction [21] and WS-Coordination [22] which defines protocols for executing Transactions among web services. [23] focuses on modeling QoS of workflows, [24] defines a QoS based middleware for services associated with the underlying workflow, but it doesn't take into account QoS factors related to Internet based services. [25] describes QoS issues related to web services from the provider's perspective. We believe that the current research has not delved into QoS issues related to Web Service based workflows, many critical issues related to the availability, reliability, performance and security of Web Services need to be handled. Our approach tactfully utilizes and monitors these QoS parameters to provide a consistent service interface to other applications in the workflow through adaptive QoS based selection, binding and execution of Web Services.

3 WebQ Framework

3.1 Architecture

WebQ framework depicted in Figure 1 is comprised of gamut of collaborative components that allows for QoS oriented adaptive management of Web Service based workflows. We first illustrate various phases of workflow management in WebQ.

Phase 1: Workflow Modelling A workflow is defined, using DAML-S [8], as a collection of tasks accomplished using in-house services-components or through third party services.

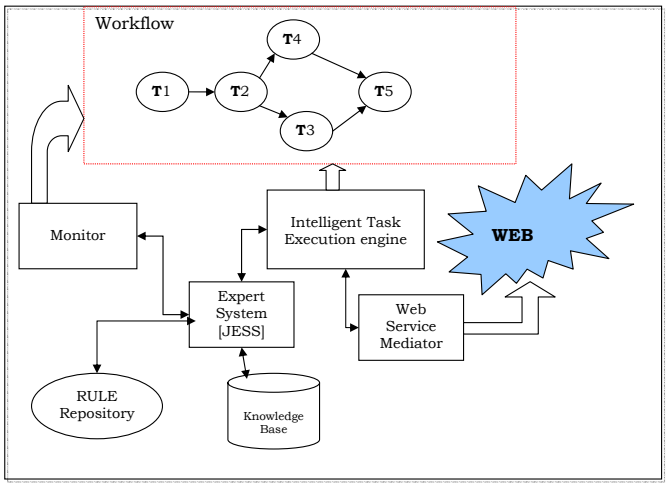

Fig. 1. The WebQ Architecture

Phase 2: QoS Requirements Setting QoS parameters associated with the underlying workflow are specified. In this paper, three categories of QoS - task-specific QoS, Internet service specific QoS and general QoS parameters were introduced. Designer specifies QoS requirements in a multilevel rule hierarchy [26].

Phase 3: Initialization WebQ fetches set of Web Services for every task in the underlying workflow. Each Web Service is assigned a uniform fitness value Ω. Fitness value Ω is defined as "How fit a web Service is for executing a task for given QoS requirement." Initially, we randomly select m services per task as every Web Service has a uniform fitness value. It could be noted here that absence of apriori knowledge about third party services calls for assignment of uniform fitness value.

Phase 4.a: Execution and Monitoring In the execution phase load (number of requests) per task is divided across m selected Web Services associated with that task. Rationale behind selecting best m Web Services and load distribution methodology is discussed in Section 3.3. Task is accomplished on execution of selected Web Services. During execution associated QoS parameters are monitored, recorded and asserted as facts in Knowledge Base.

Phase 4.b: Dynamic and adaptive Web Service selection The process of continuous monitoring of workflow under execution updates related QoS parameters which in turn updates associated fitness value. Deteriorating QoS of a selected Web Service triggers Rule that forces de-selection of that Web Service and re-selection of new set of m best services. The dynamic service selection pro-

cess accounts for the adaptability of the framework to an ad-hoc environment where quality of a Web Service changes stochastically.

In the following discussion we elicit role of each component in various phases. **Workflow**,the process under execution, is a collection of tasks that can either be accomplished by in-house services or through third party services. DAML-S [8] specification is used to specify workflow. **Web Service Mediator** queries the web for services that would accomplish a particular task. Essentially it uses information stored in UDDIs across the web to retrieve listing of task specific Web Services. **Monitor** is responsible for monitoring, measuring and asserting facts about newly calculated-observed values of general, task-specific and Internet service specific QoS parameters. It calculates various values on the basis of mathematical underpinnings of our proposed QoS model. **Intelligent Task Execution Engine** manages and co-ordinates execution of tasks that are part of underlying workflow. The engine utilizes DAML-S [8] encoded information about the workflow to co-ordinate the execution of tasks, providing input to a task, binding and execution of a Web Service associated with the task, routing the request to appropriate task depending upon the output obtained. **Expert System** allows assertion of facts in the knowledge base depending upon the rules fired. WebQ uses JESS [27] based on Rete [28], a low complexity algorithm, reduces the overhead associated with dynamic selection and binding of Web Service to a task at runtime. **Knowledge Base** is a repository of facts about Web Service related parameters - reliability, latency, execution time, performance and other QoS parameters. **Rule Repository** collects rules used to specify user specific QoS requirements, available workflow QoS and elicitation of steps to be taken for achieving specified QoS requirement. We use a multi-level approach wherein firing of a set of atomic rules leads to composite rule being executed [26].

3.2 QoS Based Service Selection and Execution

Our QoS model proposed in this paper dynamically selects the best among the available services and performs parallel execution of services. The goal of the adaptive selection and execution is to maximize the overall QoS. For the purpose, we carefully reviewed QoS parameters and classified them into the following three categories: General, Internet Service Specific and task specific QoS parameters. In order to design the Quality of Service for Web Service based workflow, multiple perspectives of the stakeholders of the system were considered. This task can be achieved by maintaining separate set of QoS management rules per user per task node, which would sum up to meet the overall desired QoS requirements.

QoS model should be flexible and extensible enough to capture the fine granularity of requirements that could arise in any given domain. [24] considers only generalized QoS features for traditional Workflow Management Systems, an Internet based Service Workflow requires a comprehensive QoS model that also incorporates task specific QoS requirements. Tables 1, 2 and 3 show a QoS model which captures all these essential features.

Table 1. General QoS parameters

QoS Parameter	*Description*	*Measure*
Performance (Latency)	The time taken to deliver services between service requestors and providers	$t_{latency} = t_{o/p}(X) - t_{i/p}(X)$, $t_{i/p}(X)$ is the timestamp when the service X is invoked and $t_{o/p}(X)$ is timestamp when the service X is delivered.
Performance (Throughput)	The number of requests served in a given period [24].	$t_{throughput}$ = Number of service invocations in time T
Reliability	This parameter is related to the number of failures of a service in a time interval.	$R = 1 - P(success)$ where $P(success)$ is as probability of successful executions, $P(success)$ = Num of successful executions/N, N: Total number of invocations
Cost	The cost of the service execution including the enactment cost (management of workflow system and monitoring [24]) and licensing fees of Web Services.	$C = C(Enactment) + C(Licensing)$

Table 2. Internet-Service Specific QoS

QoS Parameter	*Description*	*Measure*
Availability	The probability that the service will be available at some period of time. An associated parameter is time-to-repair, the time taken to repair a service [25].	$P_{availability} = C(X)/N$, $C(X)$: Num of Successful executions N: Total number of invocations $TTR = t_{restart}(X) - t_{failed}(X)$, where TTR represents Time To Repair, t_{failed} is timestamp when the service X failed, $t_{restart}$ is timestamp when service was restarted.
Security	Confidentiality, non repudiation message encryption and access control [25].	Values assigned by the workflow designer depending upon the strength of the Encryption technology used, PKI, Kerberos etc [3].
Accessibility	Instances when a particular service is not accessible even if its available because of high volume of requests.	$P_{accessibility} = P_{availability}$ at Time $T = t$
Regulatory	A quality aspect which deals with issues of conformance of service with the rules, the law, compliance with standard, and the established Service Level Agreement [25].	Specific in-house ratings by the workflow designer at design time.

Table 3. Task Specific QoS

QoS Parameter	*Description*	*Measure*
Task specific	Related to the quality of the output or the type of service offered etc.	$E(TaskQoS) = w_1 * f(p_1) + w_2 * f(p_2) + \ldots + w_n * f(p_n)$ where $f(p_i)$ refers to the probability function that maps the output of parameter p_i to a specific value depending upon how close the output is to the desired value w_i is the weight assigned to p_i.

3.3 Best One v/s Best m approach

The WebQ QoS model allows selecting best service that fits the QoS requirement. The approach of selecting a single third party service to accomplish a task is inherently unreliable. Consider a scenario where we have n available services for a given task. Our Best One QoS model approach would select one service for that task. Most of Web Services used in the workflow are third party services that cannot be trusted. It is quite possible that the execution of workflow is in jeopardy because of failure of best available service. The problem can be accounted for single point dependency between the workflow task and third party service. Our model accommodates for such problem, we advocate selection of best m against single best service. The WebQ framework allows selection of best m services based on fitness value Ω. Selection of m allows load distribution across these services. One of approach is to equally distribute load across m services. In this way better service can take more load than a average service hence a weighted average approach can distribute the load across m best services.

The load distribution function (ldf) can be expressed as a function of fitness value Ω associated with Web Service. Load per Web Service $= ldf(X_i) = L_i = N * w_i; 1 < i < m; \sum w_i = 1$, X_i is the i_{th} service from set of m best services. N is the total number of inputs at a given time t. w_i is weight associated with Web Service X_i. It is function of fitness value Ω_i of X_i. $w_i =$ f (Ω_i) A Web Service X_i with a high fitness value Ω would have high weight w_i as most of the load can be handled by better services. $w_i = \Omega_i / \sum \Omega_i$. Consider an example where N = 1000 (1000 users request weather information), $n = 5$ (5 Web Services provide weather information), $m = 3$(3 best Web Services were selected using our QoS model). Assume observed fitness values on scale of 100 as $\Omega_1 = 95$, $\Omega_2 = 35$, $\Omega_3 = 85$, $\Omega_4 = 60$, $\Omega_5 = 50$. Three best Web Services are X_1, X_3 and X_4 when top three services are selected among services arranged in descending order of fitness values. We assign weight to each Web Service $w_1 = 95/(95+85+60) = 0.395$, $w_3 = 85/(95+85+60) = 0.355$, $w_4 = 60/(95+85+60) = 0.25$. Based on this load distribution function, load per Web Service $L_1 = w_1 * N = 395, L_3 = w_3 * N = 355, L_4 = w_4 * N = 250$.

The Best m approach has a two fold advantage. First, it makes the workflow fault-tolerant. Probability of failure of workflow is now distributed across m services against a single point dependency. Failure of one or more Web Services

would not completely jeopardize the execution of underlying workflow. Second, distributing load across m services improves efficiency of overall system as similar tasks can be concurrently executed. We conclude that the concurrent, load distributed QoS model renders a workflow that is fault tolerant, efficient and reliable.

4 Implementation and Experimental Results

As a proof of concept we implemented and deployed a Java based prototype of the proposed model on Linux. Our major goal was to test the QoS management in a Web Service based workflow using JESS rule-based system. The intelligent task execution engine was constructed through Java implementation which reads the process specifications in DAML-S, selects, binds and executes Web Service from a service pool associated with each task. Monitor component asserts QoS measurements in the logDB, JESS Knowledge base. JESS Rules, measures of QoS requirements, are triggered by deteriorating Quality of Service. Dynamic service selection leads to the refinement of the workflow for maximizing the overall QoS. Java Web Services toolkit (JAX-RPC) was used for implementing the in-house Web Services and for interactions with third party Web Services.

We conducted relevant experiments to validate the proposed QoS oriented framework for adaptive management of Web Service based workflows. The tests were performed on existing services available on the Internet. We focussed on a single task in the underlying workflow. The *weather information retrieval* task was singled out as the basis of all our experiments. The choice of task was dictated by the amount of freely available third party services for that task. The main goal of the tests was to determine the performance gain by our approach of distributing load per task across m task specific services and to illustrate how the dynamic service refinement process renders an efficient, high performance and a fault tolerant workflow. Following Web Services were selected to accomplish the task of retrieving weather information.

Table 4. Input for the Performance Gain Experiments

ID	Provider[1]	WSDL
WS1	www.xmethods.net	http://www.xmethods.net/sd/2001/TemperatureService.wsdl
WS2	www.ejse.com	http://www.ejse.com/weatherservice/service.asmx?WSDL
WS3	www.juice.com	http://webservices.juice.com:4646/temperature.wsdl
WS4	FastWeather	http://ws2.serviceobjects.net/fw/FastWeather.asmx?WSDL

We tested each of the services individually, then incorporated them into our Workflow, and compared the difference in the performance. Performance gain, G is ratio of total time taken for single Web Service (n inputs) to the time

[1] The services were used for just experimental analysis, we do not intent to endorse or discriminate between providers.

taken when all services in set S are executed simultaneously with n inputs being divided between them.

We conducted tests for 10, 100, 250 and 500 inputs, corresponding performance gain was calculated, $G = 3.28, 4.31, 5.38, 5.62$ respectively. Results indicate an average performance gain of 400% ($G = 4$). Higher values of ($G > 4$) can be attributed to the fact that WebQ uses weighted average load distribution function as against uniform load distribution. The results as depicted in Figure 2a strengthens our hypotheses that the WebQ approach of load distribution renders a high performance and fault-tolerant workflow.

Secondly, we validate the proposed QoS based selection approach, a set of Web Services (depicted in Table 4) were tested rigorously across Internet and their QoS parameters were measured over a period of time. In these tests, we monitored a subset of QoS parameters - latency, throughput, and availability.

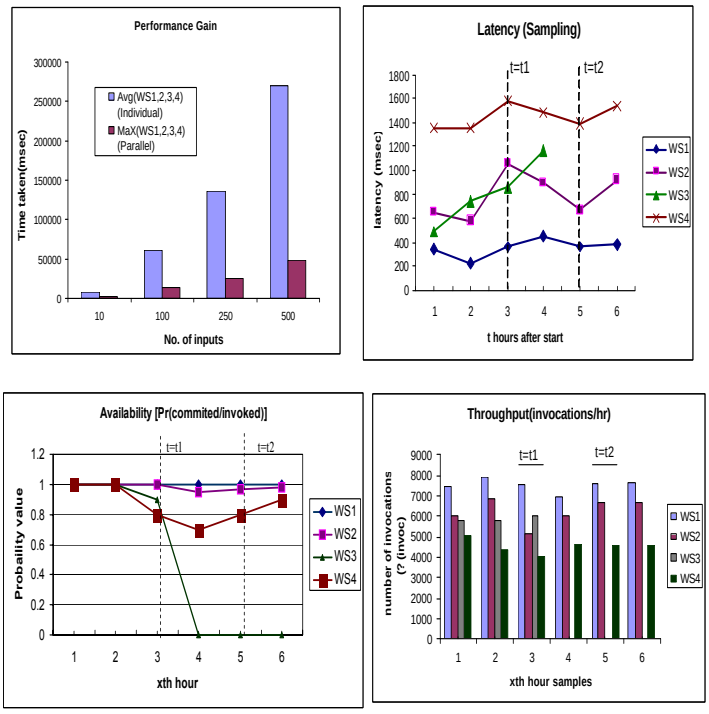

Fig. 2. Experimental Results:(a) The Performance Gain (b) Latency (c) Availability (d) Throughput

QoS requirements on selecting the task node were services which are 1) highly available 2) handle large number of inputs and 3) provide maximum throughput. The experimental results (Figure 2b, 2c, 2d) at instances $t = t_1$ and $t = t_2$

show the associated QoS parameter values for Web Services WS_1, WS_2, WS_3 and WS_4 at time t_1 and t_2 respectively. The number of inputs(load) at time $t = t_1$ were 30 and $t = t_2$ were 50. JESS Rules, measure of aforementioned QoS requirements, were triggered to select services WS_1, WS_3 (m=2) at $t = t_1$ and WS_1, $WS_2(m = 2)$ at $t = t_2$. It is important to note here that the failure of service WS_3 that was selected at time $t = t_1$ doesn't effect quality of the underlying workflow. It is because our dynamic Web Service selection at $t = t_2$ leads to selection of the new set of best m (=2) services, WS_1 and WS_2.

Moreover our research into existing Web Services revealed a highly important factor concerning the Availability parameter. Many Web Services listed on UDDIs were not available for most of time, hence we assert that Availability is an important QoS factor to take into consideration for selection of services.

Our results confirmed that Performance is also an important parameter for selection for the service. Tests revealed a large amount of variation in the performance of the Web Services. For Web Services WS_2 (mean latency = 800ms) and WS_3(mean latency = 775 ms) we observed there are very high values of variance, 110 ms and 132 ms respectively. Hence in such unpredictable Internet environment, creating a Internet based workflow demands strict performance monitoring and subsequent dynamic modification of the workflow execution.

5 Conclusions

We proposed a Web Service based framework for adaptive workflow management. A comprehensive QoS model can provide consistent interface of Web Services to workflow-based applications. Incorporated rules in the framework helped to establish a dynamic workflow by achieving a fine-grained control over service selection. The experimental results confirmed the effectiveness of the framework in enhancing the overall QoS of the system. As an ongoing work, we have extended the rule-base system to achieve automatic Web Service composition consisted of DAML-S semantics.

References

1. UDDI technical white paper, http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf, 2000
2. D. Box and D. Ehnebuske and G. Kakivaya and A. Layman and N. Mendelsohn, H. F. Nielsen and S. Thatte and D. Winer, Simple object access protocol (SOAP), www.w3.org/TR/SOAP, 2000
3. Web Services Architecture, http://www.w3.org/TR/ws-arch/
4. V. Benjamins and E. Plaza and E. Motta and D. Fensel and R. Studer and B. Wielinga and G. Schreiber and Z. Zdrahal and S. Decker, Ibrow3: An intelligent brokering service for knowledge-component reuse on the world-wide web, In The 11th Banff Knowledge Acquisition for knowledge-Based System Workshop (KAW98), Banff, Canada, 1998.
5. F. Leymann, Web services flow language, TR WSFL 1.0", IBM Software Group, May, 2001.

6. S. Thatte, XLANG Web Services for Business Process Design, 2001 http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm
7. P. Kammer, G. A. Bolcer, R. N. Taylor, M. Bergman, Techniques for Supporting Dynamic and Adaptive Workflow, Vol 9, Journal of Computer Supported Cooperative Work (CSCW), 269-292.
8. DAML-S Specifications, http://www.daml.org/services/
9. D. McDermott and M. Burstein and D. Smith, Overcoming ontology mismatches in transactions with self-describing agents, Proceedings of the First International Semantic Web Working Symposium (SWWS), 285-302, 2001
10. J. Meng, S. Y.W. Su, H. Lam and A. Helal, Achieving Dynamic Inter-organizational Workflow Management by Integrating Business Processes, Events, and Rules, Proceedings of the Thirty-Fifth Hawaii International Conference on System Sciences (HICSS-35), January 2002.
11. J. A. Miller, D. Palaniswami, A. P. Sheth, K. Kochut, and H. Singh. WebWork: METEOR 2 's web-based workflow management system. Journal of Intelligent Information Systems, 10(2):185-215, 1998
12. . J. Lee, G. Yost, and PIF Working Group. The pif process interchange format and framework. Technical Report 180, MIT Center for Coordination Science, 1995.
13. C. Schlenoff et al., The essence of the process specification language, Transactions of the Society for Computer Simulation, 16(4), pp. 204-216, 1999.
14. K. Kosanke, CIMOSA - Open System Architecture for CIM; ESPRIT Consortium AMICE, Springer-Verlag 1993.
15. E. Christensen and F. Curbera and G. Meredith and S. Weerawarana, Web services description language (WSDL) 1.1, www.w3.org/TR/wsdl, 2001
16. M. Klein and A. Bernstein, Searching for services on the semantic web using process ontologies, Proceedings of the International Semantic Web Working Symposium (SWWS), July, 2001
17. Y. Hoffner, H. Ludwig, P. Grefen and K. Aberer, Crossflow : Integrating Workflow Management and Electronic Commerce, SIGeCOM ACM 2001
18. Biztalk http://www.microsoft.com/biztalk/
19. Business Process Execution Language for Web Services, Version 1.0, July 2002 http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/
20. S. McIlraith and T. Son and H. Zeng, Semantic Web services, IEEE Intelligent Systems (Special Issue on the Semantic Web), 16(2), 46-53, 2001.
21. Web Services Transaction (WS-Transaction), August 2002, http://www-106.ibm.com/developerworks/webservices/library/ws-transpec/?dwzone=webservices
22. Web Services Coordination (WS-Coordination), August 2002, http://www-106.ibm.com/developerworks/library/ws-coor/
23. J. Cardoso, A. Sheth, J. Miller, Workflow Quality Of Service (2002)
24. A. Sheth, J. Cardoso, J. Miller, K. Koch, QoS for Service-oriented Middleware (2002) "Web Services and Grid Computing," Proceedings of the Conference on Systemics, Cybernetics and Informatics, Orlando, FL, July 2002.
25. A. Mani, A. Nagarajan, Understanding quality of service for Web services, http://www-106.ibm.com/developerworks/library/ws-quality.html
26. C. Patel, K. Supekar, Y. Lee, Adaptive Workflow Management for Web Service using QoS Framework, Technical Report TR030103, University of Missouri - Kasas City, 2003.
27. Java Expert System Shell, http://herzberg.ca.sandia.gov/jess/
28. Rete Algorithm, http://herzberg.ca.sandia.gov/jess/docs/52/rete.html