Provisioning Resilient, Adaptive Web Services-based Workflow: A Semantic Modeling Approach

Chintan Patel, Kaustubh Supekar, Yugyung Lee
School of 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 the Web. Recent efforts, OWL-S, model the semantics of Web Services that includes the capabilities of the service, the service interaction protocol, and the actual messages for service exchanges. However, there is a need to automate discovery, selection and execution of OWL-S services. Further, a framework that meets the quality of service (QoS) requirements for ad hoc Internet based services is rarely provided. In this paper, we have proposed a rule-based framework, called SemWebQ, which manages workflows composed of Semantic Web Services. SemWebQ is capable of conducting QoS-based adaptive selection as well as dynamic binding and execution of Web Services according to the semantics of workflow, thereby rendering a resilient and adaptive Web based service flow. A series of experiments performed on the SemWebQ with real Web Services have confirmed the effectiveness of proposed framework with respect to adaptive selection and execution of the Web Services in Web based workflows.

1. Introduction

Web Services have been used to provide interoperability between heterogenous applications over the Web. The Web Services standards include Web Services Description Language (WSDL), Universal Description, Discovery and Integration (UDDI) [1], and Simple Object Access Protocol (SOAP) [2]. Even though current Web Services technologies show much progress, the current services are mainly limited to atomic services. Thus, it is not adequate to handle the autonomous and complex services in realistic settings. For this problem, some research work [3] have developed languages to compose individual Web Services into transactions or workflows. Web Services Flow Language (WSFL) was designed for service compositions in a form of workflow [4], or XLANG to delineate the behavior of a single Web Services. However, these work are still short of providing adaptive Web Services for a given context.

Recently, OWL-S [5] has emerged with the objective of automatic selection, composition and interoperation of existing Web Services by a software agent. The OWL-S ontology is an agent-interpretable description of a Web Services, which adds semantics to the service profile, the process model, and the service grounding on the existing SOAP-WSDL architecture. However, too high a level of abstraction is difficult to work for Web Services invocation, composition, and monitoring of service flow. Thus, the implementation of the OWL-S specification through a pragmatic approach is required. However, there is no genuine effort to provide the automatic or semi-automatic management of service flow over the Web. Automatic management of Web Service flow requires subsequent processes of (1) discovery of Web Services considering specified constraints (2) composition of new services through automatic selection (3) execution of an identified service by an agent verification of service properties (4) execution monitoring of composite tasks performed by a set of services. A serious limitation to continued advances in automatic management of service flow is a lack of comprehensive framework for semantic modeling of services in Web based workflow.

In this paper, we enhance the Web Services based workflow modeling process using OWL-S ontology and expert system. The proposed framework, called SemWebQ, leverages the Web Services discovery, automatic selection and execution according to the OWL-S specification and QoS model. It dynamically binds the services with the underlying workflow, and performs the refinement of existing services through continuous monitoring of service execution. In order to support the workflow modeling, an OWL-S editor is developed to provide an interface for OWL-S ontology generation and visualization of the flow.

2. Related Work

Workflow technology has been around for 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, distributed workflows [7]. Process modeling languages such as IDEF, PIF, PSL and frame based models of services were used to design process typing, resource dependencies, ports, task decomposition and exception.

Current research on Web Services paves way for realizing Web based Information Systems, which have obvious advantages pertaining to scalability, heterogeneity, reuse and maintenance of services. Major issues in such a inter-organization workflow are service composition,service discovery,service contracts and the overall service quality(QoS) . 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 [4]. Service composition aids such as BizTalk [10] were proposed to overcome the limitations of traditional workflow tools which manually specify the composition of programs to perform some task. Other industrial initiatives such as BPEL4WS [11], XLANG concentrate on service representation issues to tackle the problem of service contracts, compositions and agreements. Current efforts [12] are to automate the complete process of service discovery, composition and binding using machine understandable languages. [13] focuses on modeling QoS of workflows, [14] 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.

Although our approach for Web Services composition using rule-based expert system is similar to [16], our efforts concentrate on using a OWL-S based approach, which is emerging as a widely accepted standard for modeling Semantic Web Services. Other efforts strive to achieve DAML-S based semantic matchmaking [17] of services, however, little has been to done to manage the quality of such a inter-organizational workflow that uses third party web services.This requires critical QoS management and dynamic modification of the workflow to realize a fairly stable and efficient Web Information System. Our framework provides a comprehensive solution to achieve a resilient, efficient and adaptive Web Information System. SemWebQ realizes this by employing a OWL-S Web Services model and a dynamic QoS management through a hierarchial rule-based model.

3. SemWebQ Framework

3.1. Architecture

The SemWebQ framework depicted in Figure 1 is comprised of gamut of collaborative components that render a resilient, adaptive Web based workflow. We first illustrate various phases of workflow management in SemWebQ.

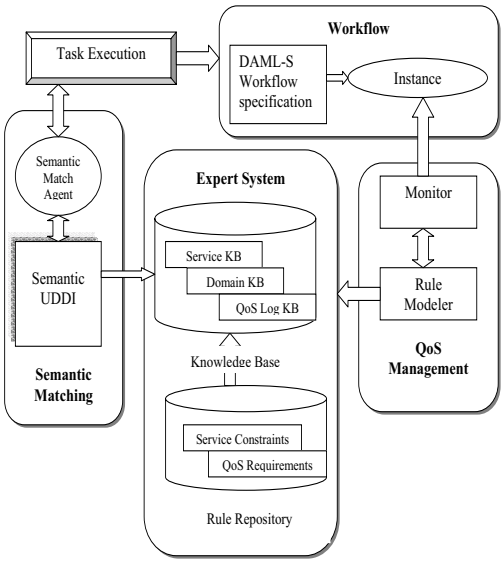

Figure 1. The SemWebQ Architecture

Phase 1: Workflow Modeling A workflow is defined, using OWL-S [5], as a collection of tasks that can either be accomplished by in-house services-components or through third party services.

Phase 2: Constraints Specification and Rule Modeling In this phase transactional and QoS constraints associated with the workflow are explicitly specified. Task specific input and output preconditions are expressed in OWL-S profile ontology, high level QoS constraints are specified by <owl:Property rdf:ID="qualityRating">. Limitations of OWL-S in expressing all the QoS requirements and service conditions demands a comprehensive model for constraint specification. In this phase, additional QoS and service specific constraints are elicited in a multilevel rule hierarchy. Section 3.3 provides a detail description of our QoS model and multilevel rule approach.

Phase 3: Initialization SemWebQ fetches set of Web Services from Semantic UDDI for every task in the underlying workflow. OWL-S ontology associated with the pre fetched Semantic Web Services is converted into JESS Knowledge base, we use DAMLJessKB[1] for the conversion. Unspeci-

fied QoS parameters are assigned a uniform value. It could be noted here that absence of apriori knowledge about third party services calls for assignment of uniform value.

Phase 4: 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 [18]. 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 5: Dynamic and adaptive Web Services Selection The process of continuous monitoring of workflow under execution updates related QoS parameters. Deteriorating QoS of the selected Web Services triggers rules that forces de-selection of that Web Services and re-selection of new set of m best services. The dynamic selection process accounts for the adaptability of the framework to an ad-hoc environment where quality of a Web Services 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. OWL-S [5] specification is used to specify workflow. **Semantic Match Agent** queries the Web for services that would accomplish a particular task. Essentially it uses information stored in Semantic UDDIs across the Web to retrieve listing of task specific Web Services. OWL-S ontologies associated with the selected Semantic Web Services are converted and stored as facts in JESS Knowledge Base. **QoS Management** ensures a QoS oriented adaptive workflow management. **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. **Rule Modeling component** allows the workflow designer to specify the QoS requirements in a multilevel rule hierarchy. **Task Execution Component** manages and co-ordinates execution of tasks that are part of underlying workflow. The component utilizes OWL-S [5] encoded information about the workflow to co-ordinate the execution of tasks, providing input to a task, binding and execution of a Web Services 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. SemWebQ uses JESS [2] based on Rete [3], a low complexity algorithm, reduces the overhead associated with dynamic selection and binding of Web Services to a task at runtime. **Knowledge Base** is a repository of facts about the Semantics and QoS of Web Services - input, output, preconditions, reliability, latency, execution time and performance. **Rule Repository** collects rules used to specify service constraints, 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.

3.2. Modeling of Semantic Web Services

We now discuss the OWL-S Ontology modeling for dynamic discovery and subsequent composition of Web Services in the workflow that imparts the desired resilient behavior. Recently, efforts are underway to develop an Ontology (OWL-S) which describes the semantics of Web Services along with the pragmatic service description which provides grounded information of the Web Services. OWL-S [5] defines a service ontology composed of (1) Service Profile, describing Web Services parameters and auxiliary features, (2) Process specifications and model, describing detailed process execution flow and (3) Service grounding with WSDL [8]. As described in [5], the process specifications for Web Services has been derived from work on standardizations of planning languages and emerging standards in process modeling and workflow technology such as PSL and Workflow Management Coalition [19]. Hence we believe that OWL-S Process Ontology meets the requirements of modeling processes in a Web-based system comprising Web Services as well as in-house components.

Although designing OWL-S Ontology is a quite challenging task, the goal of achieving automated Web Services selection, composition and execution is well worth the effort invested to build such Ontology. Furthermore, it is a challenging task to make the system capable of dynamically adjusting to the fluctuating QoS of Web services and the underlying workflow. Our solution to this problem is to use a Rule-based approach for automatic discovery, composition and subsequent execution of the service flow for given OWL-S specifications [5]. Following sections give an account of our approach of specifying Ontology and performing service matching and subsequent grounding of Web Services to execute them in the workflow.

3.2.1. Workflow Specification An abstract model of the workflow is designed using a OWL-S editor (Figure 2) by specifying required information including the pre and post conditions for the services. The editor generates the necessary OWL-S specifications describing the inputs-outputs constraints as well as the constructs for the flow control. Consider the following flow of a travelling service (refer to [18] for the details).

1 http://plan.mcs.drexel.edu/projects/legorobots/design/software/DAMLjesskb/

2 Java Expert System Shell, http://herzberg.ca.sandia.gov/jess/

3 Rete Algorithm, http://herzberg.ca.sandia.gov/jess/docs/52/rete.html

Figure 2. OWL-S Workflow Editor (a) the Workflow Designer supplies inputs-outputs, the pre-conditions and the effects of a particular task in the workflow (b) the Class Editor to define the class of pre-conditions and the effects (c) the Workflow design as viewed in the OWL-S Editor

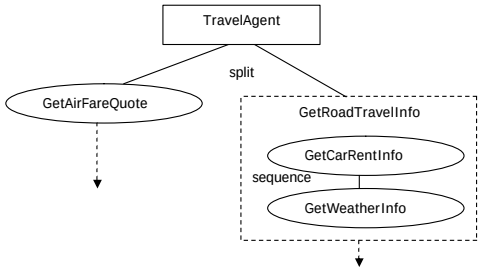

Figure 3. The Workflow Diagram

Corresponding to the designed flow (Figure 3), the editor tool will generate workflow specifications in OWL-S process ontology as given below:

<process:CompositeProcess rdf:ID="TravellersServiceFlow">
... <process:Split>...
<process:AtomicProcess rdf:about="#GetAirFareQuote" />
<process:AtomicProcess rdf:about="#GetRoadTravelInfo" />
...
</process:CompositeProcess>

TravelAgent is a composite service which splits into two processes GetAirFareQuote and GetRoadTravelInfo, which is a complex service that consists of invoking a service to calculate car rental cost for travelling by road between given locations, and then retrieves the weather information for the specified date of travel. Due to space limitation, we omit the details of the OWL-S specifications.

<process:CompositeProcess rdf:ID="GetRoadTravelInfo">
... <process:Sequence>...
<process:components rdf:parseType="Collection">
<process:AtomicProcess rdf:about="#GetCarRentInfo" />
<process:AtomicProcess rdf:about="#GetWeatherInfo" />
...
</process:CompositeProcess>

OWL-S also provides important process control constructs such as split, split-join, concurrent, choice, parallel-sync etc. beside If-then and sequence constructs.

3.2.2. Service Discovery Semantic UDDI hosts profiles of Semantic Web Services, from where the Semantic Match making Agent (SMA) can retrieve the Service ontologies. In accordance with the OWL-S specification of the given workflow, Web Services matching is defined as rules for the selection, binding and execution of the appropriate services considering various criteria: (1) Input-Output TYPE matching (2) Resolving constraints on pre and post conditions (3) Matching special service requirements (e.g., QoS from the Service Ontology <owl:Property rdf:ID="qualityRating">). We employed DAMLJessKB tool to convert the OWL-S Ontologies into knowledgebase JESS rule-based expert system to specify the corresponding rules.

(defrule matchServGetWeatherInfo
(ServKB (service ?serv)
(?INPUT $< rdfs : subClassOf >< owl : ZipCode >$)
(OR(?OUTPUT$< rdfs : subClassOf >$
$< owl : Weather >$)
(?OUTPUT $< rdfs : subClassOf >$
$< owl : Temperature >$))
(<owl:property> <owl:qualityrating> ?QOS)
(test (> QoS 3))
=> (store WEATHER ?serv)

The above mentioned rule, MatchServGetWeatherInfo is defined using JESS syntax which denotes, if a particular service has input which is subclass of class ZipCode AND output is subclass of Weather OR Temperature AND the QOS is greater that 3 (assuming star rating) then select this Web Services for the task of GetWeatherInfo. We coupled a Domain Ontology during matching process to assist the system in disambiguating concepts and terminologies thereby allowing high precision.

3.2.3. Service Grounding After the relevant service set is selected from the service discovery process, the Semantic Matching Agent (SMA)retrieves the location of the WSDL services from the OWL-S grounding specification and also maps the OWL-S parameters specified in the Ontology to the corresponding actual inputs and outputs in the WSDL.

```
(defquery WSDLServLoc
  (declare (variables ?LOCATION))
    (?serv<grounding:wsdlOperation>
          <owlsp:#GetWeatherInfo> )
    (?serv <grounding:wsdlDocument>
          ?LOCATION)
```

The above mentioned JESS query, WSDLServLoc retrieves the WSDL location for the service which performs operation defined in OWL-S process as GetWeatherInfo. Using Dynamic Java Web Services Invocation API [4] we invoke the service given the location of WSDL file.

During execution of the workflow, if a particular service fails or dwindles below the desired QoS constraints then the Semantic Match Agent (SMA) will dynamically search for another suitable matching service to complete the task without interrupting the workflow, thereby imparting resilient behavior to the workflow. Moreover, SMA can asynchronously perform a periodic search for newly available services on the UDDI, and add the service for the matching task.

3.3. QoS Modeling

3.3.1. QoS Specification The proposed QoS model dynamically selects the best among the available services and performs parallel execution of services. The goal of the dynamic, 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 (Table 1), Internet Service Specific (Table 2) and task specific QoS parameters (Table 3). In order to model the Quality of Service for Web Services 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. The QoS model should be flexible and extensible enough to capture the fine granularity of requirements that could arise in any given domain. [14] 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.

4 JavaTM Web Services Developer Pack 1.1, http://java.sun.com/webservices/webservicespack.html

3.3.2. Rule-based Modeling In SemWebQ framework, we developed a multi-level rule model (Figure 4) to achieve dynamic QoS management wherein firing of a set of atomic rules leads to composite rule being executed. The underlying motivations for the approach are (1) to handle large number of input QoS parameters, (2) to efficiently handle complex QoS requirements , (3) to allow asynchronous firing of rules to adapt changes in service requirements and (4) to dynamically change the underlying services for a task without changing existing rules.

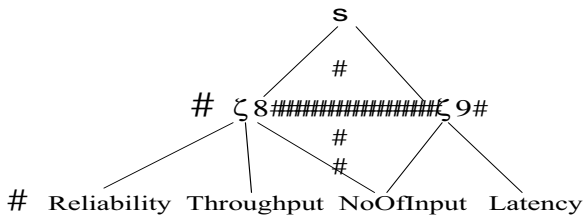

Figure 4. The Multi-level Hierarchy

First we define several terminologies used in the multi-level approach.

Definition 1: *Base* Parameter defines individual QoS parameter described in Section 3.3.

Definition 2: *Hyper* Parameter defines a combined QoS parameter denoted as symbols a_1, a_2,.... Role of the *hyper* parameter is to reduce the complexity of overall model - giving flexibility to the user. *Hyper* parameter a can be expressed as $a = f\ (w_i)\ 1 < i < N$ where f is a rule to compose parameters and N is the number of QoS parameters w_i is i_{th} *base* QoS parameter or a *hyper* parameter.

For a sample QoS requirement, two *hyper* parameters can be defined as follows: a_1 = f(throughput, availability, input size) and a_2 = f(latency, input size).

Definition 3: *Atomic* Rule composes hyper-parameter as a function of *base* parameters.

Atomic Rule AR_1:

```
(defrule selServ1
  (QoSDB (throughput ?t) (availability ?avlb) (input ?N)
  (test (> t 5000) (> ?avlb 0.8)(test (> N 20))
    => (store "a1" 0.7));
```

AR_1 describes that if the throughput ($?t$) is greater than 5000 and availability ($?avlb$) is greater than 0.8 and the number of inputs ($?N$) is greater than 20, then assign a_1 = 0.7.

Atomic Rule AR_2:

```
(defrule selServ2
  (QoSDB (latency ?l) (input ?N)
  (test (< ?l 1500)(test (< ?N 40))
  => (store "a2" 0.8));
```

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 [14].	$t_{throughput}$ = 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 [14]) and licensing fees of Web Services.	$C = C(Enactment) + C(Licensing)$

Table 1. General QoS parameters

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 [15].	$P_{availability} = C(X)/N$, $C(X)$: 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 [15].	Values assigned by the workflow designer depending upon the strength of the Encryption technology used, PKI, Kerberos etc.
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 [15].	Specific in-house ratings by the workflow designer at design time.

Table 2. Internet-Service 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.

Table 3. Task Specific QoS

AR_2 describes that if the latency ($?l$) is greater than 1500 and the number of inputs ($?N$) are greater than 40, then assign a_2 = 0.8.Using *hyper* parameters a_1 and a_2, service selection rules can be designed as follows: If ($a_1 > 0.9$) then select services that handle large number of inputs with high reliability. If ($a_1 < 0.4$) then select services which perform badly with increase in number of inputs. If ($a_2 > 0.7$) then select service that provide faster execution - lower latency for less number of inputs.

Definition 4: *Composite* Rule articulates complex rules using *hyper* parameters. A composite rule is to specify complex task specific requirements. For example, a specific QoS requirements can be expressed as a *composite* rule.

Composite Rule CR_1:
(defrule selServ3
(QoSDB (a_1 $?a_1$) (a_2 $?a_2$) (Service $?s$)
(test ($>$ a_1 0.5) ($>$ $?a_2$ 0.5))
$=>$ (store $?s$));

CR_1 expresses a QoS rule to select a service that satisfies requirements of large number of inputs with low latency and high reliability. $?s$ is the service or set of services which satisfy the given requirements.

In summary, multi-level rule approach provides flexibility in expressing complex QoS requirements by introducing *hyper* parameters and *composite* rules in combining multiple criterions in a systematic way.

4. Implementation and Experimental Results

As a proof of concept we implemented and deployed a Java based prototype of the proposed framework. Our major goals were to test the following requirements a. Generate workflow specifications in OWL-S to model complex scenarios. b. Achieve dynamic service matching and binding to the underlying workflow. c. Adaptive and resilient workflow QoS management using hierarchical rule-based approach

We developed a OWL-S Editor tool to assist the workflow designer in modeling OWL-S specifications for the workflow, which in addition provides a graphical interface to model the abstract concepts. The Service Matching Agent (SMA) handles the task of matching the Web Services for given specifications and the service matching rules as described in Section 3.2. The task execution component was constructed through Java implementation (HP Jena Toolkit[5]) which reads the process specifications in OWL-S to execute the appropriate Web Services from a service pool associated with each task. Monitor component asserts QoS measurements in the logKB - JESS Knowledge base. JESS Rules, measures of QoS requirements, are triggered by deteriorating Quality of Service. Dynamic service selection via SMA leads to the refinement of the workflow for maximizing the overall QoS. Java Web Services toolkit (JAX-RPC)[6] 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 idea of adaptiveness and resilience via rule-based QoS management of Web Service based workflows. Due to lack of availability of OWL-S based service we wrapped the existing available services on Internet with OWL-S wrappers to allow the SMA to perform service matching and binding. The tests were performed on existing services available on the Internet(Table 4). We focussed on a single task in the underlying workflow. The *GetWeatherInfo* 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 illustrate how the dynamic service refinement process renders an efficient, high performance and a fault tolerant workflow.

We validate the proposed QoS 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.

QoS requirements (specified in a multilevel rule hierarchy in Section 3.3) on selecting the task node were services which a) provide maximum throughput, b) are highly available, and 3) handle large number of inputs . The experimental results (Figure 5a, 5b, 5c) 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 Services selection at $t = t_2$ leads to selection of the new set of best m (=2) services, WS_1 and WS_2.

Our results confirmed that Performance is 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 high values of variance, 110 ms and 132 ms respectively. Hence in such unpredictable Internet environment, creating workflow demands strict performance monitoring and subsequent dynamic modification of the workflow execution.

5 HP Labs Jena 2 Toolkit, http://www.hpl.hp.com/semweb/index.html

6 JavaTM Web Services Developer Pack 1.1, http://java.sun.com/webservices/webservicespack.html

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

ID	Provider[7]	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

Table 4. Input Web Services for *GetWeatherInfo*

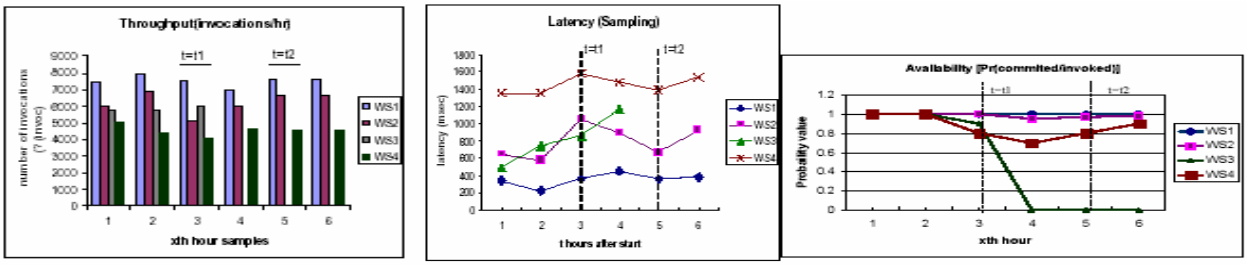

Figure 5. Experimental Results: (a) Throughput (b) Availability (c) Throughput

5. Conclusions

We proposed a semantic framework, SemWebQ, for resilient and adaptive Web Services-based workflows. The dynamic service discovery, matching and composition of the Web Services is represented using OWL-S and implemented using expert system.The results obtained from the experiments conducted with our QoS model also confirmed the effectiveness of the framework in rendering a dynamic and resilient workflow. We believe that SemWebQ framework provides a comprehensive solution towards evolving Web based Information Systems.

References

[1] UDDI technical white paper, http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf, 2000

[2] D. Box et al., Simple object access protocol (SOAP), www.w3.org/TR/SOAP, 2000

[3] V. Benjamins et al., 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.

[4] F. Leymann, Web Services Flow Language, TR WSFL 1.0, IBM Software Group, May, 2001.

[5] OWL-S Specifications, http://www.daml.org/services/

[6] 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.

[7] 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

[8] E. Christensen and F. Curbera and G. Meredith and S. Weerawarana, Web services description language (WSDL) 1.1, www.w3.org/TR/wsdl, 2001

[9] M. Klein and A. Bernstein, Searching for services on the semantic web using process ontologies, Proceedings of the International Semantic Web Working Symposium, 2001

[10] Biztalk http://www.microsoft.com/biztalk/

[11] Business Process Execution Language for Web Services, Version 1.0, July 2002 http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[12] 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.

[13] J. Cardoso, A. Sheth, J. Miller, Workflow Quality Of Service (2002)

[14] 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, 2002.

[15] A. Mani, A. Nagarajan, Understanding quality of service for Web services, http://www-106.ibm.com/developerworks/library/ws-quality.html

[16] S. R. Ponnekanti and A. Fox. SWORD: A Developer Toolkit for Web Service Composition. The Eleventh World Wide Web Conference, 2002.

[17] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, Katia P. Sycara: Semantic Matching of Web Services Capabilities. International Semantic Web Conference 2002: 333-347

[18] C. Patel, K. Supekar, Y. Lee, Adaptive Workflow Management for Web Service using QoS Framework, Technical Report TR030103, University of Missouri - Kansas City, 2003.

[19] Workflow Management Coalition, http://www.aiim.org/wfmc.