Research Journal of Applied Sciences, Engineering and Technology 7(19): 4072-4078, 2014
DOI:10.19026/rjaset.7.769
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2014 Maxwell Scientific Publication Corp.
Submitted: December 04, 2013 Accepted: December 17, 2013 Published: May 15, 2014

Research Article

QoS Aware Adaptive Service Composition

[1]A. Chitra and [2]M. Nageswara Guptha
[1]Department of Computer Applications,
[2]Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, Tamilnadu, India

Abstract: In current business scenario the business processes of enterprises are service oriented. The services of various applications are advertised in service registry using Web Service Description Language description for service selection and composition. Service composition process integrates services based on Business Process Execution Language description of composite service and executed by the orchestration coordinator. As all compliant services description are included in the description, the service selection and execution consumes more time users need select the service form set of complaint services. This study proposes an adaptive service composition methodology over extended web service Quality Aware Service Integration architecture (QASI). The proposed service composition methodology reads the composite service description and constructs a composition tree. In each service node of composition tree a service is replaced by one that satisfies functional and QoS requirement. During execution of composite service or binding of service, the parameters like server maximum capacity, current load on server and expected time for completing current load for service at each level of composition tree are analyzed. An alternative service will be allocated at particular level to improve the overall response time of composite service. The QASI architecture provides a repository to store composition tree as symbolic notations to ensure repeatability. Experimental results indicates selection of best service and better response time by proposed methodology using QASI architecture compared to contemporary web service composition methodology using static BPEL description.

Keywords: Distributed application development, QoS aware service integration, service composition, service integration, service oriented computing

INTRODUCTION

Service Oriented Computing (SOC) is inevitable in business processes with ever changing requirements. Basic building block of SOC is services which are accessed through internet. The services are discovered and invoked using open XML standards like WSDL (Web Service Description Language), SOAP (Simple Object Access Protocol) and UDDI [Universal Description Discovery and Integration] (Thomas, 2007).

Services from various service providers are described in service registry using WSDL. Most of the services do not meet the requirement of business applications. To accomplish business process requirements, it is required to combine various individual services and their sequence of execution defined.

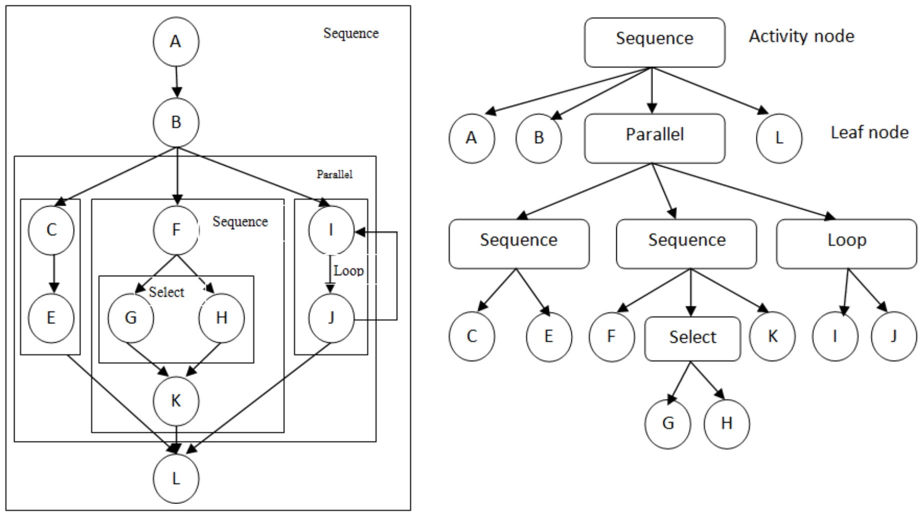
Service composition: Number of reusable elementary services are combined together to form a composite service. A composite service contains n tasks in sequence (i^{th} level task [l_i] will be executed after i-1^{th} level task [l_{i-1}]) or in parallel (all same level tasks will be executed simultaneously) or in switch (either i^{th} or j^{th} node of same level task will be executed) or loop (the i^{th} level task will be executed n times) depending on execution flow.

Service composition can be realized using two approaches:

- Orchestration
- Choreography

Orchestration defines entire process in top-down mode. The BPEL description of the orchestration, the orchestrated code, is used by controller to execute various services. Unlike orchestration, in choreography the interaction is peer-to-peer that is, one service will initiate other service that are involved in business process. The key difference between these two is that the orchestration approach assumes a single central point of control over the entire scope of the process execution, while the choreography approach assumes

Corresponding Author: A. Chitra, Department of Computer Applications, PSG College of Technology, Coimbatore, Tamilnadu, India



(a) Composite service for a business process (b) Composition tree

Fig. 1: Composition tree

Table 1: QoS attributes

S. No	QoS attribute name	Description	Unit
1.	Response time	Maximum or average time taken to send a request and receive a response	ms
2.	Availability	Number of successful invocations/total invocations	%
3.	Throughput	Total Number of invocations for a given period of time	Invokes /sec
4	Successability	Number of response/number of request messages	%
5.	Reliability	Ratio of the number of error messages to total messages	%
6.	Latency	Time taken for the server to process a given request	ms

that execution control is shared, potentially across multiple business processes (Thomas, 2007).

The six major issues of composition of services are Coordination, transaction, context, conversation model, Execution monitoring and infrastructure (Schahram and Wolfgang, 2005).

Service composition can be classified as following:

- Static (or design time) and Dynamic (or run time) service composition.
- Automated and Manual service composition.

The process of service composition requires an effective method to facilitate quick and simple composition of services and remains as a key challenge to realize the full potential of services.

Composition tree: The composition process constructs composition tree using Business Process Execution Language (BPEL) description of composite service (Big *et al*., 2009). The construction of composition tree is illustrated in Fig. 1. The business process flow contains activity nodes and nodes '*A*' to '*J*' represent services in Fig. 1a. The tree representation of composite service is shown Fig. 1b.

QoS of composite service: The service registry contains a number of services with same functionality. These services may differ in nonfunctional or QoS attributes like response time, latency and cost. To improve the efficiency of composite service, the candidate services are ranked and the best service is selected for each leaf node in composition tree according to constraints specified by consumer. A subset of QoS attributes for service composition is shown in Table 1.

Existing QoS aware service composition methodologies select the service from service registry based on user functional and QoS requirement. The selected services are allocated to respective leaf nodes of composition tree. For service selection only, a subset of QoS is considered. It is also noted that all structural activities are not accounted while computing QoS of composite service.

This study proposes an adaptive QoS aware service composition methodology over extended web service architecture QASIA to address the issues studied. The proposed system scans the BPEL description of composite service and converts into a composition tree. To improve the quality of composite service a better alternative is discovered from service registry using QoS parameters and this replaces the existing service. The resulting composition tree is converted into symbols and stored in Service sequence registry.

During service binding, the benefit value of server is calculated based on server maximum load, its current load on server and execution time for completion of current load. If benefit value exceeds the threshold, then the service will be included in the composition else the second best service is selected from service registry in order to improve the execution time.

The proposed system is experimented with benchmark web service dataset (http://www.uoguelph.ca/~qmahmoud/qws/dataset/) for service discovery to select best service based on QoS parameter using multi utility ranking function and found that it automatically discovers the best quality alternative from the compliant services.

LITERATURE REVIEW

Many service composition systems that are designed to solve well defined problems lack flexibility for volatile user requirements. Layered framework for service composition and aggregation (Sun *et al*., 2003; Pires *et al*., 2003; Guptha *et al*., 2009) integrates services based on user functional requirements and generates static composition sequence for composite service. However it does not provide infrastructure to preserve and modify the composite service.

A mapping model to map the mathematical or abstract model and the BPEL description using business rules and composition rules is described (Orriens *et al*., 2003a, b). This mapping model does not include nonfunctional requirements of composite service.

Ran's experiment concludes that in the conventional service registry, 40% of UDDI entries are unusable because of inaccurate information. Further service discovery in registry is limited to functional requirements (Ran, 2003).

Dynamic service composition was proposed (Casati and Shan, 2001). An Integrated Development Environment for semantic based dynamic service composition designed to build composition plan manually, creates a static business plan using BPEL (Chatle *et al*., 2007; Aniss *et al*., 2008). It allows the user to select the service to satisfy the business requirement and to modify the plan. The signature information of services is placed in service registry along with basic and QoS information. The user has to provide all parameters to access a service for composition. A framework is designed to discover the service based of behavior (Bensheng, 2010).

An optimized service composition was proposed by Bing *et al*. (2009). This is applicable for sequence flow of service composition while dynamic QoS parameters like server capacity are not considered for QoS computation (Bing *et al*., 2009). Heuristics and genetic algorithms are used for QoS aware service composition (Berner *et al*., 2006; Canfora *et al*., 2005).

The ontology based service composition uses the description of services provided in service registry for service composition (Ying, 2011; Aabhas *et al*., 2011). This results in incompatible service composition if output parameter type of a service does not match with input parameter of its successor.

The existing composition methodologies use either semantic information or message types for service composition, which would combine incompatible services. The service discovery process of these systems may results in poor quality of composite service. During service discovery, the request should contain values for all attributes like name or ID of service, input and output parameters with their types, category of service and values for QoS parameter. Based on the request the existing service discovery process linearly searches all service entries available in service registry to select and allocate service at each level of composition sequence which makes discovery and composition processes time-consuming.

PROPOSED SYSTEM

The proposed system constructs a composite tree based on composite service description and discovers better alternative service using adaptive service composition methodology over QASI architecture.

QoS Aware Service Integration Architecture (QASIA): QASI architecture extends the conventional web service infrastructure by providing repository to

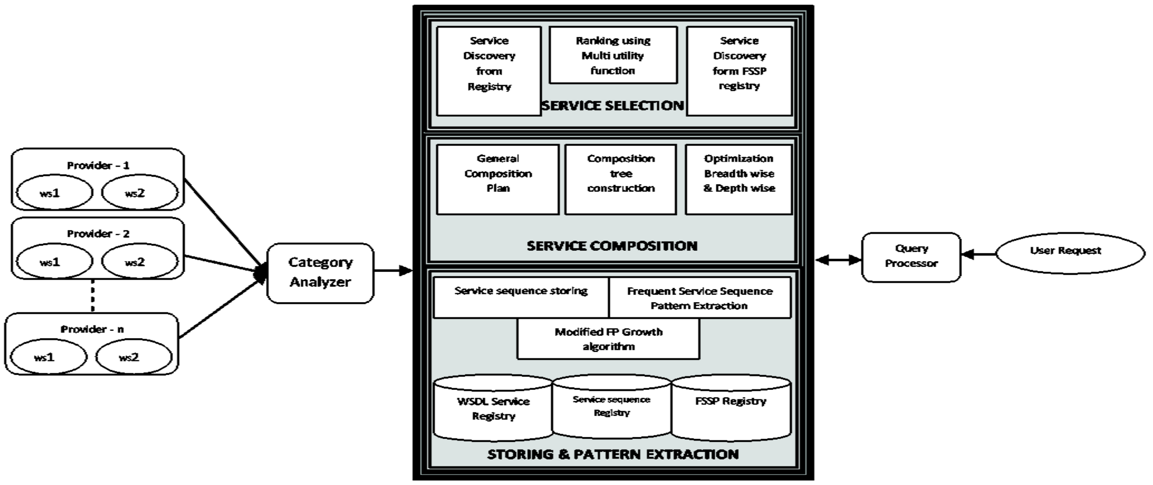

Service Discovery from Registry
Ranking using Multi utility function
Service Discovery form FSSP registry
SERVICE SELECTION
General Composition Plan
Composition tree construction
Optimization Breadth wise & Depth wise
SERVICE COMPOSITION
Service sequence storing
Frequent Service Sequence Pattern Extraction
Modified FP Growth algorithm
WSDL Service Registry
Service sequence Registry
FSSP Registry
STORING & PATTERN EXTRACTION
Provider - 1
ws1
ws2
Provider - 2
ws1
ws2
Provider - n
ws1
ws2
Category Analyzer
Query Processor
User Request

Fig. 2: QoS Aware Service Integration Architecture [QASIA]

Table 2: Composition tree construction

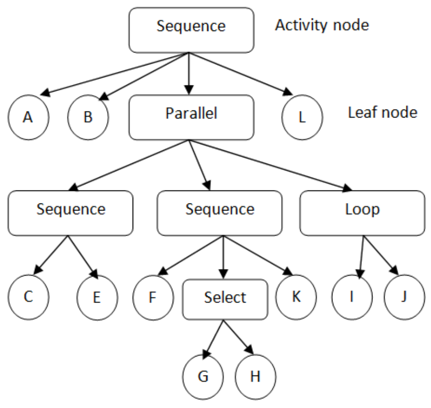
BPEL description Composition for business process of Fig. 1a	Composition tree
< SEQUENCE> <invoke>sA<\ invoke > < invoke >sB<\ invoke > <PARALLEL> < SEQUENCE> < invoke >sC<\ invoke > < invoke >sE<\ invoke > <\SEQUENCE> < SEQUENCE> < invoke >sF<\ invoke > < SELECT> < invoke >sG<\ invoke > < invoke >sH<\ invoke > <\SELECT> < invoke >sK<\ invoke > <\SEQUENCE> < LOOP> < invoke >sI<\ invoke > < invoke >sJ<\ invoke > <\LOOP> <\PARALLEL> < invoke >sL<\ invoke > <\SEQUENCE>	 Sequence Activity node A B Parallel L Leaf node Sequence Sequence Loop C E F Select K I J G H

Table 3: Contents of composition sequence registry

Composition sequence
rSE(*l1sA*:*l1sB*:*l1PL*(*l2SE*(*l3sC*:*l3sE*):*l2SE*(*l3sF*:*l3SL*(*l4sG*:*l4sH*) :*l3sK*):*l2LP*(*l3sI*:*l3sJ*)):*l1sL*)

S: Service, *r*: Root level node, l_i: Level of node (Where i = 1….n), *SE*: Sequence activity node (all services is to be executed linearly), *PL*: Parallel activity node (all services is to be executed simultaneously), *SL*: Select activity node (Subject to condition any one service is to be executed), *LP*: Loop activity node (all services is to be executed repeatedly), :-To concatenate sibling nodes, ( ): Enclose siblings

store composition sequence (Fig. 2). It assumes that the service vendors submit their service in a common service registry. The service registry is classified into number of classes based on end utility of service. Each class has a collection of service providing same functionalities and their QoS values.

Composite tree construction: The tree construction process scans the composite BPEL description and search for BPEL tags like *sequence*, *flow*, *switch*, *while* and *pick* which depict the sequence of execution of services. For such tags, activity nodes are created and added to composition tree. Within these activity tags, *invoke* tags are provided to call required services. For each *invoke* tag a service node will be added as children to respective activity node and a unique ID is assigned for each service node and same is used to identify the corresponding service Table 2.

The constructed composition tree is converted into symbols and stored in composition sequence registry as shown in Table 3.

Service selection process: The service nodes of composition tree are analyzed and similar services are extracted from service registry. The selected services along with existing service are ranked via multi utility ranking function.

Multi Utility Ranking (MUR): The compliant services are ranked using the QoS values of the respective service and replace the existing service of composition tree. The utility value of QoS attributes is calculated using Eq. (1):

$$Q_{new}(A_i) = \frac{1}{fre} \times fn\big(P(A_i)\big) + \big(w_i \times fn\big(R(A_i)\big)\big) \qquad (1)$$

where,

fre	=	Number of times the service is used in particular duration (Eg. Per day)
$P(A_i)$	=	Quality rating value of attribute provided by the service provider
$R(A_i)$	=	Average quality rating value of attribute perceived by the previous service instances measured using benchmark tools
w_i	=	Weight of the QoS attribute given by current service user
Fn	=	Any normalization function or distance vector function

The QoS for a service is calculated using Eq. (2):

$$TQ\big(s_j\big) = \sum_{i=1}^{n} Q_{new}\,(A_i) \qquad (2)$$

The QoS attributes like Average response time, Latency are to be minimized and attributes like Availability, Throughput are to be maximized. The overall rating for service is calculated using multi utility Eq. (3) and (4):

Table 4: QoS formula for activity nodes

QoS property	Sequence	Parallel	Select	Loop
Service cost	$\sum_{i=1}^{n} \mathrm{sc}(\mathrm{sl_i})$	$\sum_{i=1}^{n} \mathrm{sc}(\mathrm{sl_i})$	$\sum_{i=1}^{n} \mathrm{sc}(\mathrm{sl_i}) \times \mathrm{r}(\mathrm{sl_i})$	n x sc(sl$_i$)
Average response time	$\sum_{i=1}^{n} \mathrm{rt}(\mathrm{sl_i})$	$max_{i=1}^{n}\mathrm{rt}(\mathrm{sl_i})$	$max_{i=1}^{n}\mathrm{rt}(\mathrm{sl_i})$	n x rt(sl$_i$)
Throughput	$min_{i=1}^{n}\mathrm{tpt}(\mathrm{sl_i})$	$\sum_{i=1}^{n} \mathrm{tpt}(\mathrm{sl_i})$	$\sum_{i=1}^{n} \mathrm{tpt}(\mathrm{sl_i})$	tpt(sl$_i$)
Successability	$\prod_{i=1}^{n} \mathrm{suc}(\mathrm{sl_i})$	$\prod_{i=1}^{n} \mathrm{suc}(\mathrm{sl_i})$	$\sum_{i=1}^{n} \mathrm{suc}(\mathrm{sl_i}) \times \mathrm{r}(\mathrm{sl_i})$	suc(sl$_i$)
Reliability	$\prod_{i=1}^{n} rbt(\mathrm{sl_i})$	$\prod_{i=1}^{n} rbt(\mathrm{sl_i})$	$\sum_{i=1}^{n} \mathrm{rbt}(\mathrm{sl_i}) \times \mathrm{r}(\mathrm{sl_i})$	rbt(sl$_i$)
Latency	$\sum_{i=1}^{n} \mathrm{lat}(\mathrm{sl_i})$	$max_{i=1}^{n} lat(\mathrm{sl_i})$	$\sum_{i=1}^{n} \mathrm{lat}(\mathrm{sl_i}) \times \mathrm{r}(\mathrm{sl_i})$	n x lat(sl$_i$)

sl$_i$: Service at level *i*, sc(*sl$_i$*): Service cost of level *i* service, *rt*(*sl$_i$*): Averageresponse time of level i service, *tpt*(*sl$_i$*): Throughput of level i service, *rbt*(*sl$_i$*): Reliabilityof level i service, suc(*sl$_i$*): Successabilityof level *i* service, *lat*(*sl$_i$*): Latencyof level i service, n: Number of leaf nodes of activity node, *r*(*sl$_i$*): Rate at which the level *i* service is selected

$$MTQ(s_{j_{max}}) = \sum_{i=1}^{n} Q_{new}(A_{i_{max}}) \quad (3)$$

$$MTQ(s_{j_{min}}) = \sum_{i=1}^{n} Q_{new}(A_{i_{min}}) \quad (4)$$

The service ranked first is allocated to respective leaf node. The runner service is stored temporarily for replacement of the allocated service during execution if server load of binding server is high.

QoS computing for activity node of composition tree: The activity node of composition tree is classified into four namely Sequence, Parallel, Select and Loop. Table 4 describes the QoS for each type of activity.

The overall QoS of composite service is computed by summing of QoS values of all activity nodes involved in composition tree.

Dynamic service selection: The services that are selected for composition are executed in respective server where its definition is available. The service binding is decided by benefit function of server if greater than threshold and if less than the time taken to replace the current service. The Benefit function *Bfn* is shown in Eq. (5):

$$Bfn = \frac{C_{max}(\mathrm{s}) - C_{curr}(\mathrm{s})}{C_{max}(\mathrm{s})} \quad (5)$$

where,

C_{max}(S) = Maximum load on server (Mostly 100)

C_{curr}(S) = Current load on server (in PHP sys_getloadavg ())

EXPERIMENTAL RESULTS

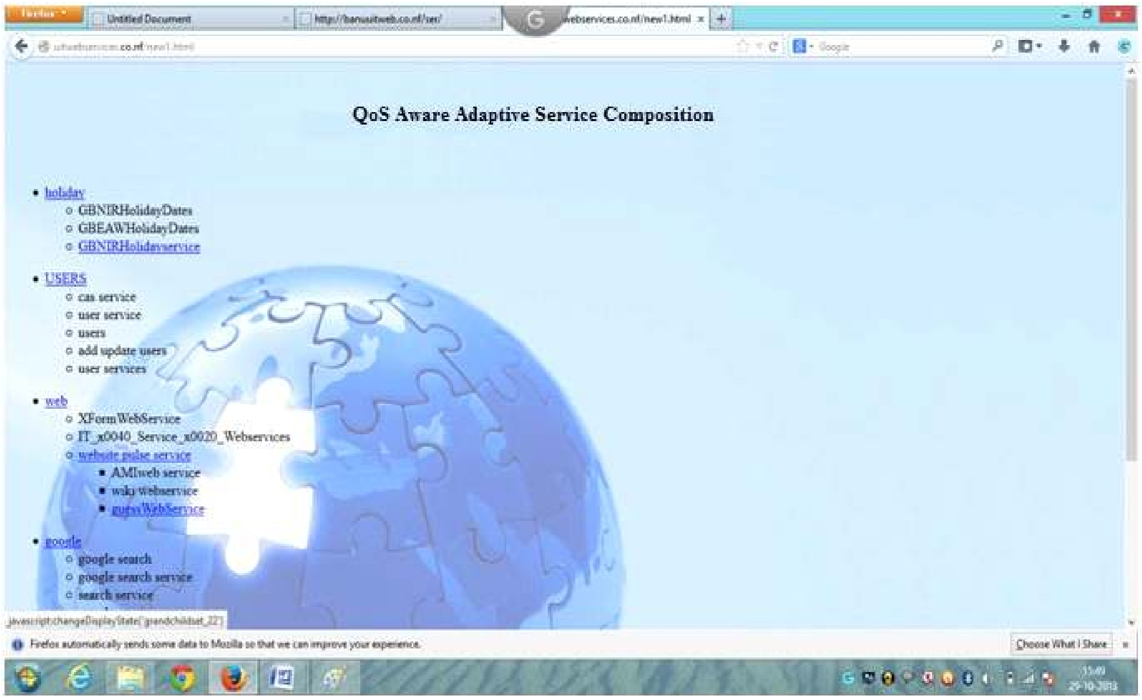
The proposed system has been experimented by implementing service shown Table 5 Experimental data is taken from Quality of Web Services [QWS] Dataset with 2500 services (Mohammad and Thomass, 2009), ("http://www.uoguelph.ca/~qmahmoud/qws/ dataset").

The services of QWS dataset are grouped under classes-Vehicle class, Travel class, Math class, Enterprise class, which are further classified into sub classes. For instance vehicle class has subclasses like Vehicle sales, Vehicle purchase and Vehicle repair service (Fig. 3).

An initial composition plan with 15 services involving all activities like sequence, select, loop and parallel is described using BPEL and composition tree is constructed. The constructed composition tree contains 15 service nodes and four activity nodes. A web service for composition tree construction is implemented which transforms the composite service description into a composition tree. The QoS attribute values of service in created composition tree are extracted and pooled with respective compliant service values in a multi dimensional matrix. Ranking service selects the best and second best service for each service node of composition tree. The created composition tree is converted in symbolic notation and stored along with service IDs along with respective binding URL in a relational database. Snapshot for service discovery for News service is shown in Fig. 4. The system computes the mean for selected QoS attributes and considers this mean as threshold if it is not provided by the user. Similarly the user can provide weight for each QoS attribute as shown in Fig. 5.

Table 5: Services implemented for experimentation

Name of service	Description
Composition Tree construction service	Constructs composition tree using BPEL description
Service selection and allocation service	To discover compliant services from service registry
Control service	Controls sequence of execution of all mutually exclusive services available in orchestrated code.
Ranking service (QoS computing)	Ranks the selected services using Multi-utility ranking



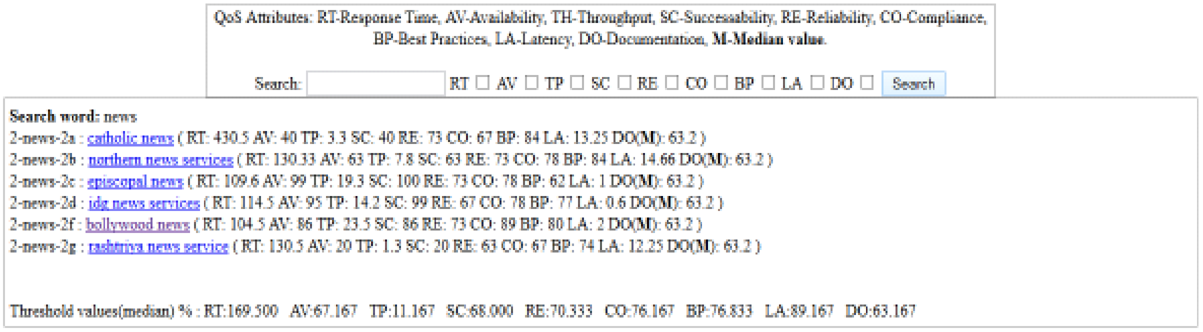
QoS Aware Adaptive Service Composition

- holiday
  - GBNIRHolidayDates
  - GBEAWHolidayDates
  - GBNIRHolidayservice
- USERS
  - cas service
  - user service
  - users
  - add update users
  - user services
- web
  - XFormWebService
  - IT_x0040_Service_x0020_Webservices
  - website pulse service
    - AMIweb service
    - wiki webservice
    - guessWebService
- google
  - google search
  - google search service
  - search service

javascript:changeDisplayState('grandchildset_22')

Firefox automatically sends some data to Mozilla so that we can improve your experience.

Fig. 3: Classification of services



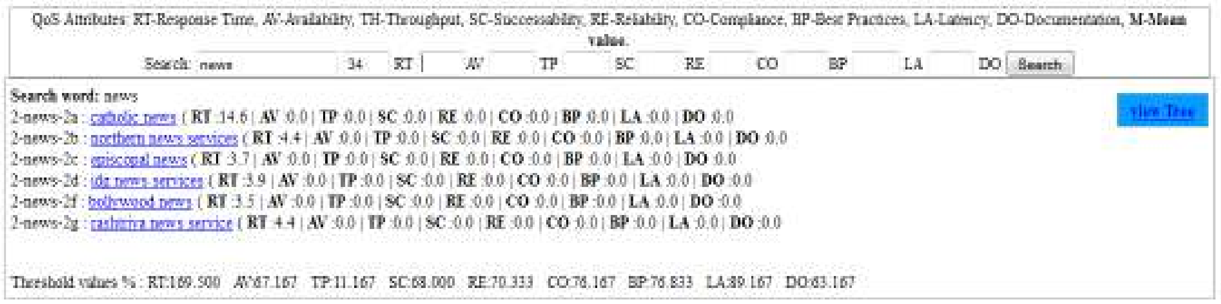
QoS Attributes: RT-Response Time, AV-Availability, TH-Throughput, SC-Successability, RE-Reliability, CO-Compliance, BP-Best Practices, LA-Latency, DO-Documentation, **M-Median value.**

Search: RT ☐ AV ☐ TP ☐ SC ☐ RE ☐ CO ☐ BP ☐ LA ☐ DO ☐ Search

Search word: news
2-news-2a : catholic news (RT: 430.5 AV: 40 TP: 3.3 SC: 40 RE: 73 CO: 67 BP: 84 LA: 13.25 DO(M): 63.2)
2-news-2b : northern news services (RT: 130.33 AV: 63 TP: 7.8 SC: 63 RE: 73 CO: 78 BP: 84 LA: 14.66 DO(M): 63.2)
2-news-2c : episcopal news (RT: 109.6 AV: 99 TP: 19.3 SC: 100 RE: 73 CO: 78 BP: 62 LA: 1 DO(M): 63.2)
2-news-2d : idg news services (RT: 114.5 AV: 95 TP: 14.2 SC: 99 RE: 67 CO: 78 BP: 77 LA: 0.6 DO(M): 63.2)
2-news-2f : bollywood news (RT: 104.5 AV: 86 TP: 23.5 SC: 86 RE: 73 CO: 89 BP: 80 LA: 2 DO(M): 63.2)
2-news-2g : rashtriya news service (RT: 130.5 AV: 20 TP: 1.3 SC: 20 RE: 63 CO: 67 BP: 74 LA: 12.25 DO(M): 63.2)

Threshold values(median) % : RT:169.500 AV:67.167 TP:11.167 SC:68.000 RE:70.333 CO:76.167 BP:76.833 LA:89.167 DO:63.167

Fig. 4: Searching for news service with QoS attributes



QoS Attributes: RT-Response Time, AV-Availability, TH-Throughput, SC-Successability, RE-Reliability, CO-Compliance, BP-Best Practices, LA-Latency, DO-Documentation, **M-Mean value.**

Search: news 34 RT AV TP SC RE CO BP LA DO Search

Search word: news
2-news-2a : catholic news (**RT** :14.6 | **AV** :0.0 | **TP** :0.0 | **SC** :0.0 | **RE** :0.0 | **CO** :0.0 | **BP** :0.0 | **LA** :0.0 | **DO** :0.0
2-news-2b : northern news services (**RT** :4.4 | **AV** :0.0 | **TP** :0.0 | **SC** :0.0 | **RE** :0.0 | **CO** :0.0 | **BP** :0.0 | **LA** :0.0 | **DO** :0.0
2-news-2c : episcopal news (**RT** :3.7 | **AV** :0.0 | **TP** :0.0 | **SC** :0.0 | **RE** :0.0 | **CO** :0.0 | **BP** :0.0 | **LA** :0.0 | **DO** :0.0
2-news-2d : idg news services (**RT** :3.9 | **AV** :0.0 | **TP** :0.0 | **SC** :0.0 | **RE** :0.0 | **CO** :0.0 | **BP** :0.0 | **LA** :0.0 | **DO** :0.0
2-news-2f : bollywood news (**RT** :3.5 | **AV** :0.0 | **TP** :0.0 | **SC** :0.0 | **RE** :0.0 | **CO** :0.0 | **BP** :0.0 | **LA** :0.0 | **DO** :0.0
2-news-2g : rashtriya news service (**RT** :4.4 | **AV** :0.0 | **TP** :0.0 | **SC** :0.0 | **RE** :0.0 | **CO** :0.0 | **BP** :0.0 | **LA** :0.0 | **DO** :0.0

Threshold values % : RT:169.500 AV:67.167 TP:11.167 SC:68.000 RE:70.333 CO:76.167 BP:76.833 LA:89.167 DO:63.167

Fig. 5: Searching for news service with QoS attributes and weight

The system receives a new request from user and verifies the composition sequence repository. The matching composition sequence is extracted and appropriate services are called using orchestration coordinator rather creating new composition description using BPEL. It is also easy to replicate the composition sequence in various geographical locations to increase availability. The space consumed by the composition sequence is relatively less than the BPEL description of composite service.

The proposed service composition methodology for adaptive QoS aware service composition selects relatively best service using QoS parameters than the conventional methodology that uses static BPEL description.

CONCLUSION

The proposed QoS aware Service Composition has been designed and implemented over QASI architecture. This architecture ensures Coordination among participating services, Transaction control, Execution monitoring, Reusability of quality services and extends infrastructure for caching service composition sequence. The services for composition tree are selected from service registry using multi utility ranking function and replaced if necessary. The system dynamically selects an alternative service during execution using server capacity of binding server. The proposed QoS aware Service composition system is experimentally found to provide relatively better quality composite service than conventional methods using static BPEL description in service discovery and service composition.

REFERENCES

Aabhas, V.P., S. Basit, V. Jaideep, X. Hui and A. Nabil, 2011. Semantic-based automated service discovery. IEEE T. Serv. Comput., 5(2): 260-275.

Aniss, A., M. Hafedh and O. Abdel, 2008. Signature-based composition of web services. Proceeding of the International MCETECH Conference on e-Technologies, pp: 104-114.

Bensheng, Y., 2010. A new framework for web service discovery based on behavior. Proceeding of the IEEE Asia Pacific Services Computing Conference, pp: 654-658.

Berner, R., M. Spahn, N. Repp, O. Heckmann and R. Steinmetz, 2006. Heuristics for QoS-aware web service composition. Proceeding of the IEEE International Conference on Web Services (ICWS, 06).

Bing, L., S. Yuliang and W. Haiyang, 2009. QoS oriented web service composition and optimization in SOA. Proceeding of the Joint Conference on Pervasive Computing, pp: 605-610.

Canfora, G., M.D. Penta, R. Esposito and M.L. Villani, 2005. An approach for QoS-aware service composition based on genetic algorithms. Proceeding of the Genetic and Evolutionary Computation Conference (GECCO, 2005). Washington, DC, USA.

Casati, F. and M.C. Shan, 2001. Dynamic and adaptive composition of E-services. Proceeding of the 12th International Conference on Advanced Information System.

Chatle, G., G. Das, K. Dasgupta, A. Kumar and S. Mittal, 2007. An integrated development environment for web service composition. Proceeding of the IEEE International Conference on Web Services (ICWS, 2007). Salt Lake City, UT, pp: 839-847.

Guptha, M. N., A. Chitra and P.T. Rajan, 2009. Three tier architecture for service oriented business intelligence. Int. J. Inform. Process., 3(4): 67-76.

Mohammad, A. and R. Thomass, 2009. Combining global optimization with local selection for efficient QoS-aware service composition. Proceeding of the 18th International World Wide Web Conference (WWW 2009). Madrid, Spain.

Orriens, B., J. Yang and M.P. Papazoglou, 2003a. Model Driven Service Composition. In: Orlowska, M.E. (Ed.), ICSOC, 2003. LNCS 2910, Springer-Verlag, Berlin, Heidelberg, pp: 75-90.

Orriens, B., J. Yang and M.P. Papazoglou, 2003b. A Framework for Business Rule Driven Web Service Composition. In: Jeusfeld, M.A. and O. Pastor (Eds.), ER 2003 Workshop. LNCS 2814, Springer-Verlag, Berlin, Heidelberg, pp: 52-64.

Pires, P.F., M.R.F. Benedives and M. Mattoso, 2003. Building Reliable Web Services Composition. In: Chaudhri, A.B. *et al*. (Eds.), Web Databases and Web Services, LNCS 2593, Springer-Verlag, Berlin, Heidelberg, pp: 59-72.

Ran, S., 2003. A model for web service discovery with QoS. ACM SIGecom Exch., 4(1): 1-10.

Schahram, D. and S. Wolfgang, 2005. A survey on web service composition. Int. J. Web Grid Serv., 1(1).

Sun, H., X. Wang, B. Zhou and P. Zou, 2003. Research and Implementation of Dynamic Web Services Composition. In: Zhou, X. (Ed.), APPT 2003. LNCS 2834, Springer-Verlag, Berlin, Heidelberg, pp: 457-466.

Thomas, E., 2007. Service-oriented Architecture Concepts, Technology and Design. Pearson Education, Prentice Hall, pp: 760.

Ying, Z., 2011. Semantic-based data and service unified discovery. Proceeding of the 4th International Joint Conference on Computational Sciences and Optimization, pp: 787-791.