Architecture for Evaluating Web Service QoS Parameters using Agents

M.Thirumaran
Department of Computer Science and Engineering, Pondicherry Engineering College

P.Dhavachelvan
Department of Computer Science and Engineering, Pondicherry University

S.Abarna
Department of Computer Science and Engineering, Pondicherry Engineering College

G.Aranganayagi
Department of computer Science and Engineering Pondicherry Engineering College

ABSTRACT
As Web services have become the grand vision these days, more and more people are seeking out the practicalities of implementing and using them for business benefit. Thus Web services make application functionality available over the Internet in a standardized, programmatic way. QoS support for Web service has become a widely researched area and has shown to be an effective mechanism in Web services' discovery particularly in differentiating between services that share similar functionalities and finally by evaluating QOS and providing interface for selecting the web service. In this paper, we are providing a sophisticated architecture for quality driven web service evaluation. Agents are used to evaluate the QWS parameters. This work also discussed about the quality attributes with organized set of design related questions which helps an evaluator to analyze the ability of the architecture to meet quality requirements, and provides a brief sample evaluation. ATAM method of software architecture evaluation is used to evaluate the proposed model. The assessment justifies the proposal in terms of the performance attributes such as reliability, availability, modifiability, security and interoperability etc.

Keywords: Web services, QWS parameters, agents, software architecture, ATAM

1. INTRODUCTION

Web services are considered as self-contained, self describing, modular applications that can be published, located, and invoked across the Web. Nowadays, many companies and organizations implement their core business and application services over Internet. Thus, the ability to efficiently and effectively select and integrate inter-organizational and heterogeneous services on the Web at runtime is an important step towards the development of the Web service applications [2]. A large number of web services are being developed as an emerging standard to construct distributed applications in the web. Service requesters have access to a choice of descriptions to various services that provide similar service functionality. Automation of dynamic web service discovery is made viable by expression of domain specific knowledge [3] [4]. Service discovery is to match service requirement and service capability. Service requirement is originated from service consumers who want to complete Internet-based tasks. They hope to use complex but flexible search mechanism to get exact and needed services [5].

If multiple Web services provide the same functionality, then a Quality of Service (QoS) requirement can be used as a secondary criterion for service selection. QoS is a set of non-functional attributes like service response time, throughput, reliability, and availability [6] [7]. The current Universal Description, Discovery and Integration (UDDI) registries only support Web services discovery based on the functional aspects of services [6]. The problem, therefore, is firstly to accommodate the QoS information in the UDDI, and secondly to guarantee some extent of authenticity of the published QoS information. QoS information published by the service providers may not always be accurate and up-to-date. To validate QoS promises made by providers, we propose that consumers rate the various QoS attributes of the Web services they use. These ratings are then published to provide new customers with valuable information that can be used to rank services for selection. Web service QoS reputation can be considered as an aggregation of QoS ratings for a service from consumers over a specific period of time.

This provides a general estimate of the reliability of a service provider. With service reputation taken into consideration, the probability of finding the best service can be increased. However, the assumption is that the customer ratings are considered non-malicious and fairly accurate.

Therefore, only semantic ranking is not enough, and other nonfunctional properties of services such as price, reputation and reliability should be computed and ranked. Unfortunately, although QoS-based service selection and ranking have been a hot topic research area [8][9], it's hard to come up with a standard QoS model that can be used for all services in all domains. This is because QoS is a broad concept that can encompass a number of context-dependent nonfunctional properties. Moreover, when evaluating QoS of web services, we should also take into consideration domain specific criteria [10]. Since QoS computing and evaluating become very important in the presence of multiple grid services with overlapping or identical functionality .By considering all these above stated facts, a robust architecture is proposed to automatically evaluate the QWS parameters to ensure quality driven web service discovery.

In section 2, we described proposed system, QWS parameter evaluation by agents, explained working of each component and in section 3, we evaluated the proposed architecture for quality driven web service discovery and in Section 4 analysis the architecture with scenarios and its

attributes. Section 5 states the conclusions and Section 6 lists the references.

2. LITERARY SURVEY

Web services used primarily as a means for businesses to communicate with each other and with clients, Web services allow organizations to communicate data without intimate knowledge of each other's IT systems behind the firewall. Unlike traditional client/server models, such as a Web server/Web page system, Web services do not provide the user with a GUI. Web services instead share business logic, data and processes through a programmatic interface across a network. The applications interface, not the users. Developers can then add the Web service to a GUI (such as a Web page or an executable program) to offer specific functionality to users. Web services allow different applications from different sources to communicate with each other without time-consuming custom coding, and because all communication is in XML, Web services are not tied to any one operating system or programming language. For example, Java can talk with Perl, Windows applications can talk with UNIX applications. This is made possible by using technologies such as Jini, UPnP, SLP, etc.

Slim Trabelsi and Yves Roudier proposed a scalable solution to enabling secure and decentralized discovery protocols. It also deals how to extend the WS-Discovery Web Service protocol with these mechanisms [11]. Colin Atkinson and Philipp Bostan proposed the brokerage aspect of the web service vision but it is difficult to involve in setting up and maintaining useful repositories of web services. So they describe a pragmatic approach to web service brokerage based on automated indexing and discuss the required technological foundations [12]. Janette Hicks and Weiyi Meng proposed a current discovery research through use of the Google Web service, UDDI category searching and private registry. They found WSDL documents for a given domain name, parse the desired service document to obtain invocation formats, and automatically invoke the Web service to support enhancements of HTML-dependent search tools by providing access to data inaccessible through surface HTML interfaces [13]. ZHANG Changyou and ZHU Dongfeng invented a web service discovery mechanism on unstructured P2P network. The web services are clustered into communities through functional properties and several query packets will be proliferated and spread through the community. Each service in this community will be evaluated through non-functional properties. The service clustering and experience exchanging enhanced the efficiency in discovery [14]. Henry Song and Doreen Cheng examine better approaches of using general-purpose search engines to discover Web Services. They used Yahoo and Google search engine and the queries were fired to each search engine daily and the top 100 search results returned from every search are collected and analyzed. The results show that for both search engines, embedding a WSDL specification in a Web page that provides semantic description of the service [15].

3. PROPOSED SYSTEM

We have proposed architecture for Quality driven Web service discovery which allows for exact service discovery for composite process and satisfies accurately user's specific requirements. External user interacts with the user agent by submitting their requests through Query component. This component then submits the request to the Request Analyzer, where the request is parsed with respect to the type of the request, existing or new. Existing query is serviced by retrieving the information from the database of Service patterns which are dynamically updated automatically and by the processing new requests. The new query is passed to the Semantic Analyzer where in the component extracts the meaning into keywords which best depicts the query and makes a search in the UDDI registries.

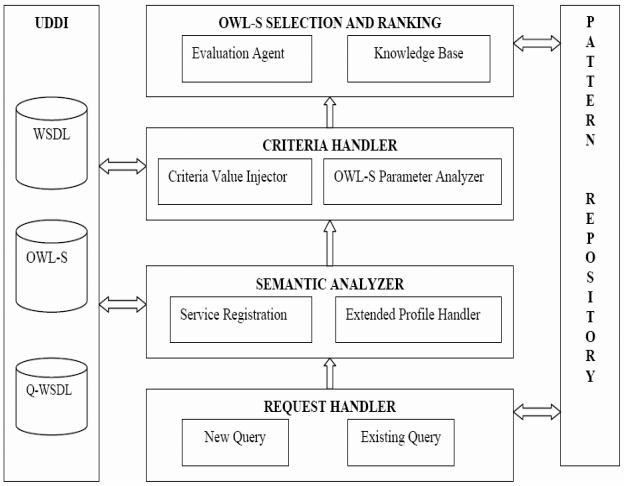

Figure 1(a). Architecture for Web Service Discovery

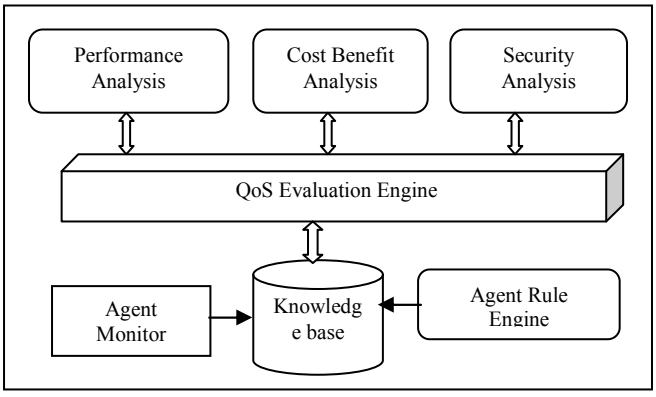

Figure 1(b). Architecture for Web Service evaluation

The UDDI registries enable businesses to publish service listings and discover each other and define how the services or software applications interact over the Internet. A UDDI business registration consists of three components: White Pages - address, contact, and known identifiers; Yellow Pages- industrial categorizations based on standard taxonomies; Green Pages - technical information about services exposed by the business. The web service publishers publish WSDL files

consisting of above information regarding the services offered by the businesses. These are stored in the WSDL repositories. Criteria value Injector adds special tags into the WSDL files which are already published by information provided by the QWS parameter analyzer. QWS parameter analyzer is the vital component of the architecture where all the evaluation of the QWS parameters is done. The parameters are classified into special criteria like performance, reliability, security, usability etc. The evaluation is done making use of the information from server log files, certifier, service log file, service usage pattern, service profile and feedback. The ranking of the web services is done with the help of evaluation agent and knowledge base and a list of web services meeting the client's criteria is provided among which the client can make a choice of the web services. This prioritized list is stored in the service pattern database for future use if same kind of request is made.

The QOS evaluation engine of figure 1(b) evaluates various QWS parameters like response time arability, throughput, reliability, best practices etc using Application Manger .further the evaluated QOS parameters are categorized and submitted to performances, cost, and security analyzers. The agent monitoring systems evaluates and stores the QWS parameters in agent knowledge based then agent rule engine facilitate to select the best services based on the QOS requirements. This evaluation engine acts and interfaced as Evaluation Agent in the architecture of evaluating QOS parameters in figure 1(a).

4. EVALUATION OF PROPOSED ARCHITECTURE

The proposed architecture is evaluated by the Architecture Tradeoff Analysis Method (ATAM) [12], [13], [14]. All the scenarios corresponding to each applications of the service mining are listed and evaluated based on quality attributes. Before evaluation, we have to identify the different stakeholders involved in the system. The Stakeholders are developers, maintainers, evaluation team, customer representative, architecture team, business analysts, end-user, operator, tester, system administrator. The evaluation team presents ATAM to above stated stakeholders with brief explanation of steps and techniques followed for analyzing and eliciting utility tree generations, architectural approaches and scenario mapping and result of evaluation identified with stakeholders prioritize, risks, tradeoff, response, response measure.

Next, goal of architecture is identified and based on the analysis of stakeholders need and present the business goals. The utility tree provides a mapping between the quality attributes that the architecture to meet discussed in business driver to the corresponding scenarios. In this tree, root node is "utility" and second level node are various quality attributes of architecture and third level follows attribute concerns and in final fourth level represents scenario with pair of ranking to represent the prioritize of nodes in leaves. The scenarios are prioritized relative to each other using ranking pairs of (High, Medium, and Low). It would be (H, H) (H, M) (H, L) (M, H) (M, M) (M, L) (L, H) (L, M) (L, L). The first letter denotes degree of importance to system and second letter denotes degree of difficulty in achieving it. The scenario prioritization is included in table 2.

According to SEI [12], the suitability of the architecture proposed is determined by quality attribute requirements that are important to stakeholders of system. The ATAM relies on elicitation of quality attribute scenarios. The Scenarios chosen for evaluation of the architecture are given below in the table 1. The Scenario 1 and Scenario15 comes under reliability. Scenario2, Scenario6, Scenario7, Scenario8, Scenario9, Scenario10, Scenario13 and Scenario18 comes under performance. Scenario 11, Scenario14 and Scenario17 based on the security attributes. Scenario3, Scenario4, Scenario5, Scenario12, Scenario16 comes under extensibility.

Table 1. Scenario Identification

SCENARIO No	SCENARIOS	QUALITY ATTRIBUTES
Scenario 1	User request is processed double(including duplicate request)	Reliability
Scenario 2	Response time of the system should be less	Performance
Scenario 3	New service provider with own service interface is added	Interoperability
Scenario 4	New architecture component /business partner is added	Modifiability
Scenario 5	Ability to accommodate new requirements	Adaptability
Scenario 6	Provider fails after advertise service	Availability
Scenario 7	Increase number of users for a service	Scalability
Scenario 8	Ability to accommodate new technology	Maintainability
Scenario 9	Communicate execution status/Common look and feel GUI	Usability
Scenario 10	Policies of services changes or modified	Flexibility
Scenario 11	Payment processing functions should be secure	Security
Scenario 12	User request is satisfied by discovering a correct service	Correctness
Scenario 13	Three concurrent requests for a service came	Capacity
Scenario 14	Intruder trying to access the resources	Security
Scenario 15	A data exception occurs and the system notifies a defined list of recipients by mentioning offending conditions in red on data screens	Reliability
Scenario 16	User wants the information regarding classes of the desired web service ,which route It takes and interoperable services	Manageability
Scenario 17	User who is not registered is trying to login to access web services	Security (authentication)
scenario18	Minimize the latency to 100 ms.	Performance

5. ARCHITECTURE ANALYSIS

Architecture Analysis is to reveal enough information about the proposed architecture to identify it risks, non-risks, tradeoffs, sensitive points in the design phase itself rather than later phase. This method is not meant with precise and detailed evaluation of architecture quality attributes with its numerical value[12]. In the evaluation phase, we identified risks associated with architectural decisions and their effects on quality attributes. Table 2 shows with defined scenarios and their risks, tradeoff, scenario prioritization. The Architecture Tradeoff Analysis Method (ATAM) reveals that proposed architecture for service mining agent with information's like risk, tradeoff points, response measure, and priority in design phase. The most important tradeoff and sensitive points identified in proposed system is highly sensitive to performance of the system and in next phase of implementation care should be taken to control sensitiveness of system performance. In addition to the tradeoffs

and sensitive points, several risks associated with architecture are also identified using ATAM.

Table 2. Analysis of Scenarios in Architecture

QUALITY ATTRIBUTES	SCENA -RIOS	GOALS	RISKS	TRADEOFFS	RESPONSE MEASURE	PRIORITY
Reliability	S1	User satisfaction	-	Provides Idempotent end points-Performance overhead and complexity in Implementation	100% Transaction	(H,M)
Performance	S2	User satisfaction	No control over execution of External service providers	Provide Interoperability and loose coupling-latency of request increases	System replies to user with in few seconds	(H,H)
Interoperability	S3	Ease of integration	-	Interact with external service interface-extensive XML processing-Performance overhead	Not more than 10 person per day	(H,H)
Modifiability	S4	Ease of Upgradation	-	-	Not more than 1 person per day	(M,L)
Adaptability	S5	User satisfaction	-	Wireless communication-may affect reliability	Not more than 5person per day	(L,M)
Availability	S6	Provide user reliable service	No control to prevent the authorized provider to advertise service	Monitoring providers and checks for consumer time's out-Performance and Reliability Overhead	Not more than 10 person per day	(H,H)
Scalability	S7	User satisfaction	-	Centralized DB to handle multiple instances-poor Response time	Not more than 1person per day	(H,H)
Maintainability	S8	Ease of Upgradation	Less impact in system	-	1 Spiral	(L,H)
Usability	S9	User satisfaction	-	Increases User's learn ability of system-attackers enters and security of system is in critical	Not more than 3 person per day	(H,M)
Flexibility	S10	Provider satisfaction	-	Monitoring changes in Providers functionality and updates in DB-performance Overhead	Not more than 1person per day	(H,H)
Security	S11	Assure security	Digital certificate is done with care	Adds performance overhead and adds complexity in implementing digital certificate authentication	Not more than 15 person per day	(H,M)
Correctness	S12	User satisfaction	Interface are too finely grained	Adds performance overhead in transmission &processing of many small messages to complete a task	Not more than 3 person per day	(H,M)
Capacity	S13	User satisfaction	too many users will make the architecture complex	performance overhead, depends on how intelligent the agent is	-	(H,H)
Security	S14	Assure security	-	there is a chance of blocking of real user thus reducing user satisfaction	Not more than 15 person per day	(H,M)
Reliability	S15	provide Comfort to user	In processing many requests leads to a chance in reducing user satisfaction	there may be a chance in reducing user satisfaction	System replies to user with in few seconds	(H,M)
Manageability	S16	User satisfaction	No control over providing the reliable information to	Performance overhead in doing extra work than providing service	Not more than 3 persons per day	(M,H)

5. IMPLEMENTATION

We implemented for banking application using netbeans with GlassfishServerV2. Here we have created web service, LoanApproval for approving loan based on some criteria's such as qualification, age, property details, purpose etc.., Also it maintains the customer profile after approving the loan. Another web service, InterestCalculator Service calculates interest based on loan type. It uses the user profile created by the LoanApproval Service and finally calculates interest for the loan. The QoS such as computability, traceability, accessibility is evaluated using Application Manager 7. After the evaluation is completed a graph is generated based on the evaluation using Application Manager 7.

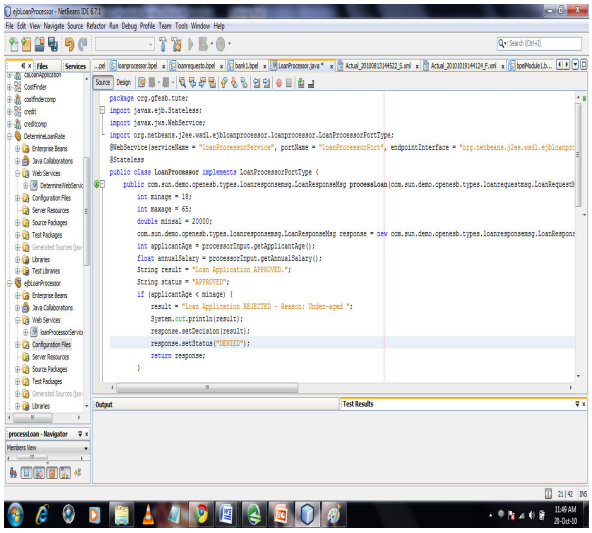

Figure 2. LoanApproval Service

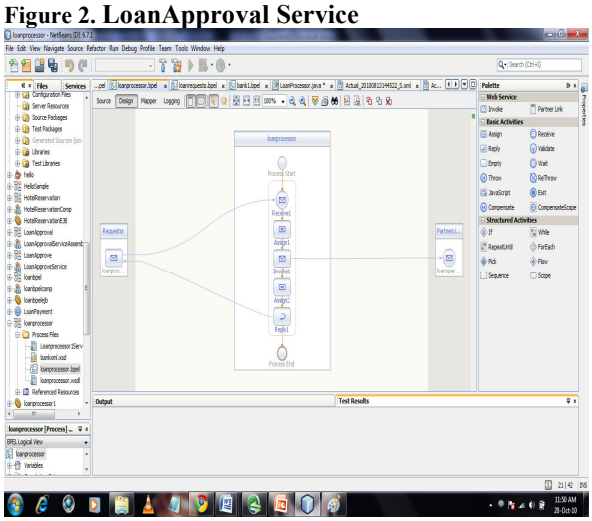

Figure 3. InterestCalculator Service

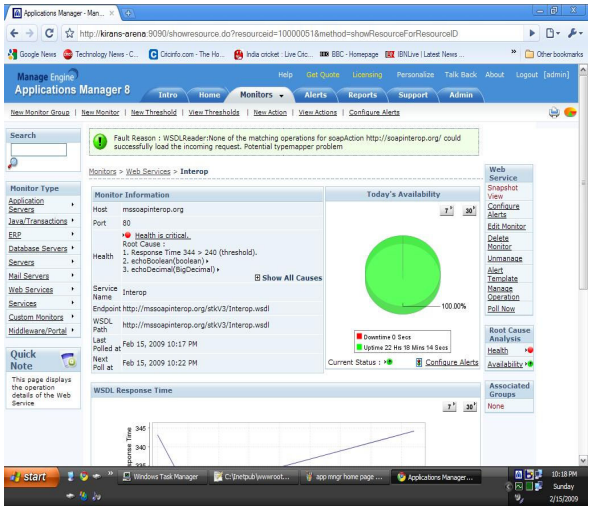

Figure 4. QoS Evaluation using Application Manager 7.

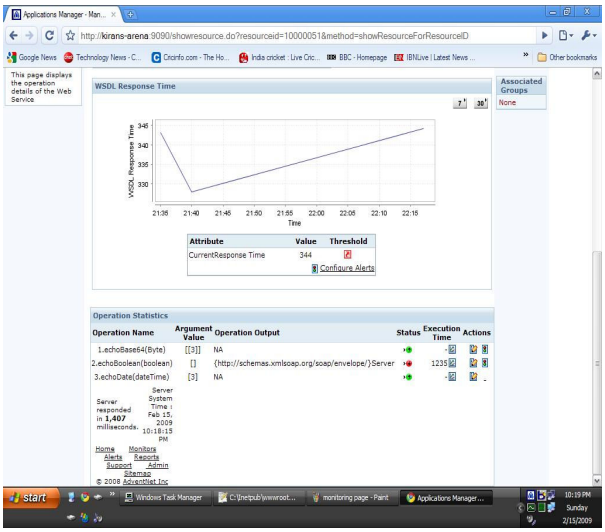

Figure 5. QoS Evaluation graph using Application Manager 7.

QoS Evaluation

Calculation of performance:

- For zero millisecond = 100 points for every 10ms(x) = 1 point deduction.
  Gross points (GP) = (100-((response time)/x))*weight). (Weight of Response time =5 & for latency=4)
- Throughput GP= (Throughput / Maximum value of Throughput)*100 % *weight. (Here Max value = 27.2, weigh of throughput = 3)
- Capacity GP = % of capacity * weight. (Weight of Capacity = 2)

Calculation of Request Satisfaction:

- Availability GP = (1 - (Down Time/Measurement time)) 100% * weight
- Accessibility GP = (Number of acknowledgements received / Total number of requests)*100%* weight.
- Successibility GP= (Number of response messages / Number of request messages)*100%*weight.
- Usability is taken as the, (average percentage of Accessibility and Successibilty) * weight.
- Weights for Availability, Accessibility, Successibility and Usability are 5, 4, 3 &2 respectively.

Calculation of Reliability:

- WsRF is taken as the (overall performance percentage based on the values of Response time,
- Latency, Throughput and Capacity) *weight.
- Service Reputation GP = ((No. of invocations / Maximum no. of invocations)*100%)*weight. Here Maximum value = 79
- Weights: Robustness = 5; WsRF = 4; WsRF = 4; Service Reputation = 3.

Calculation of Manageability and Security:

- Weights: manageability = 2; security = 1.
- Assuming four levels in Manageability and Security:
  - High (90-100)
  - Medium (75-89)
  - Low (50-74)
  - Worst (0-49)

The following tables show the calculation of QWS parameters using the QoS Evaluation tool Application Manager 7.

Table 3: Criteria - Performance

Web Services	Availability GP	Accessibility GP	Successibility GP	Usability GP	Overall GP	Criteria Rank
DictionaryService	360	312	168	134	974	3
MyService	390	260	183	126	959	4
Aba	410	304	192	140	1046	1
AlexaWebSearch	335	252	168	119	874	6
Errormailer	310	188	162	101	761	10
Getjoke	435	236	168	115	954	5
Fastweather	415	260	201	132	1008	2
XigniteRetirement	395	224	135	101	855	7
DOTSEmailValidate	270	272	153	119	814	9
XigniteNews	305	256	156	116	833	8

6. CONCLUSION

In this paper we addressed actual need of a robust and standard architecture for an efficient web service discovery to meet client's requirements evaluating the QoS parameters providing quality driven web services. The various steps in the architecture development phases are explained in this paper. The proposed architecture is evaluated using Architecture Tradeoff Analysis Method (ATAM) which allows identifying risks, non-risks, sensitive points, tradeoffs, priority of each scenario to system previously in design phase.

7. REFERENCES

[1] Liang, Q.A., Miller, S., Chung, J.-Y., 2005. "Service mining for Web service composition", IEEE International Conference on. Information Reuse and Integration2005, Volume, Issue, 15-17 Page(s): 470 – 475, Aug. 2005.

[2] Qianhui Liang, Jen-Yao Chung, Steven Miller, Ouyang Yang, 2006." Service Pattern Discovery of Web Service Mining in Web Service Registry-Repository", IEEE International Conference on E-Business Engineering, Pages: 286-293, 2006.

[3] Qianhui Liang J. Y. Chung, 2007 "Analyzing Service Usage Patterns: Methodology and Simulation" IEEE International Conference on e-Business Engineering (ICEBE'07) pp. 359-362.

[4] Zheng, George, Bouguettaya, Athman, July 2007. "A Web Service Mining Framework", IEEE International Conference on Web Services, Volume, Issue, 9-13 Page(s):1096 – 1103.

[5] Mohsen Rouached , Walid Gaaloul , Wil M. P. van der Aalst , Sami Bhiri and Claude Godart,"Web Service Mining and Verification of Properties: An Approach Based on Event Calculus", Cooperative Information Systems (CoopIS) 2006 International Conference, Springer Berlin / Heidelberg, Volume 4275/2006, Pages 408-425,2006.

[6] Walid Gaaloul, Sami Bhiri, Claude Godart, 2006. "Research Challenges and Opportunities in Web Services Mining" ICWS'06 September 2006.

[7] Schahram Dustdar, Robert Gombotz,2006." Discovering web service workflows using web services interaction mining", International Journal of Business Process Integration and Management ,Vol. 1, No.4 pp. 256 - 266.

[8] Chhabra, Manish Lu, Hongen La Trobe University, Australia, 2007."Towards Agent Based Web Service ", International Conference on Computer and Information Science, On page(s): 93-99.

[9] Bin Li; Xiao-Yan Tang; Jian Lv, 2005 ."The Research and Implematation of Services Discovery Agent in Web Services Composition Framework " Proceedings of 2005 International Conference on Machine Learning and Cybernetics, Volume 1, Issue , 18-21 Page(s): 78 – 84, Aug. 2005.

[10] Vinoski, S.2003." Service discovery 101", IEEE Internet Computing Volume 7, Issue 1, Page(s): 69 – 71 Jan/Feb 2003.

[11] Slim Trabelsi, Jean-Christphe Pazzaglia and Yves Roudier, "Secure Web service discovery: overcoming challenges of ubiquitous computing" The European Conference on Web Services, ECOWS'06.

[12] Colin Atkinson and Philipp Bostan,"A Practical Approach to Web Service Discovery and Retrieval" IEEE International Conference on Web Services, ICWS 2007.

[13] Janette Hicks, Madhusudhan Govindaraju, Weiyi Meng, "Search Algorithms for Discovery of Web Services" IEEE International Conference on Web Services, ICWS 2007.

[14] ZHANG Changyou1 and ZHU Dongfeng, "A Web Service Discovery Mechanism Based on Immune Communication" International Conference on Convergence Information Technology, 2007.

[15] Henry Song and Doreen Cheng, "Web Service Discovery Using General Purpose Search Engines" IEEE International Conference on Web Services, ICWS 2007.