ISSN 2319-7595

International Journal of Information Systems and Computer Sciences, Vol.7. No.1, Pages : 16-22 (2017)
Special Issue of ICSIC 2017 - Held during 23-24 September 2017 in Amman Arab University, Amman-Jordan
http://www.warse.org/IJISCS/static/pdf/Issue/icsic2017sp16.pdf

Dynamic Service Oriented Architecture Quality of Service Driven

Prof. Khalid Kaabneh
Department of Computer Science, College of Computer Science and Informatics.
Amman Arab University
kaabneh@aau.edu.jo

Dr. Hassan Tarawneh
Department of Mobile Computing, College of Computer Science and Informatics.
Amman Arab University
hassan@aau.edu.jo

Dr. Issam Al-Hadid
Department of Business Information Technology, College of Information Technology.
Jordan University
i.hadid@ju.edu.jo

***Abstract*--Service-oriented architecture (SOA) middleware has emerged as a powerful and popular distributed computing paradigm due to its high-level abstractions for composing systems and encapsulating platform-level details and complexities. Control of some details encapsulated by SOA middleware is necessary, however, to provide managed quality-of-service (QoS) for SOA systems that require predictable performance and behavior. This paper presents a policy-driven approach for managing QoS in SOA systems called QoS Enabled Dissemination (QED). QED includes services for (1) specifying and enforcing the QoS preferences of individual clients, (2) mediating and aggregating QoS management on behalf of competing users, and (3) shaping information exchange to improve real-time performance. We describe QED's QoS services and mechanisms in the context of managing QoS for a set of Publish-Subscribe-Query information management services. These services provide a representative case study in which CPU and network bottlenecks can occur, client QoS preferences can conflict, and system-level QoS requirements are based on higher level, aggregate end-to-end goals. We also discuss the design of several key QoS services and describe how QED's policy-driven approach bridges users to the underlying middleware and enables QoS control based on rich and meaningful context descriptions, including users, data types, client preferences, and information characteristics. In addition, we present experimental results that quantify the improved control, differentiation, and client-level QoS enabled by QED.**

Keywords*-- *Service Oriented Architecture, Quality of Service, Information Management.

I. INTRODUCTION

Web services technology increasingly has been used to develop the new software systems' era, by moving from module implementation to unit composition, which is the base of the Service Oriented Architecture (SOA). Mariani [1] stated *"Customers demand for high quality systems that typically require a large amount of time to be developed and must be released in stable versions"*.

Web service technology can reduce the time to market, as well as the Quality of Service (QoS) according to the Service Agreement Level (SAL) must be provided by the service providers that can gain the clients' reputation and increase the market share. The Quality of Service (QoS) of a Web Service may be affected either because of internal changes or because of workload fluctuations. Guaranty the Quality of service requirements (availability, accessibility, integrity, reliability, throughput, latency, Security). Yu and Lin [2] proposed a method that promises to offer a better quality of service; it uses QoS constraints to choose the service and binds it.

SOA consists of 3 levels [3, 21, 13],:

1- Service Level: Web services are the base of the Service Oriented Architecture, so we need a mechanism to make sure that these units provide the expected behavior.
2- Communication level: the traffic between Web service providers and requesters.
3- Flow Level (Business Process Execution): the Web Service business process execution.

Ben Halima et al. [3] proposed an architectural framework for monitoring and analysis of Web Services' QoS driven by models for QoS analysis. Their suggested framework used to monitor and analysis the SOAP messages between Web services. Figure 1, shows their architectural framework applied to a food shop scenario.

ISSN 2319-7595

International Journal of Information Systems and Computer Sciences, Vol.7. No.1, Pages : 16-22 (2017)
Special Issue of ICSIC 2017 - Held during 23-24 September 2017 in Amman Arab University, Amman-Jordan
http://www.warse.org/IJISCS/static/pdf/Issue/icsic2017sp16.pdf

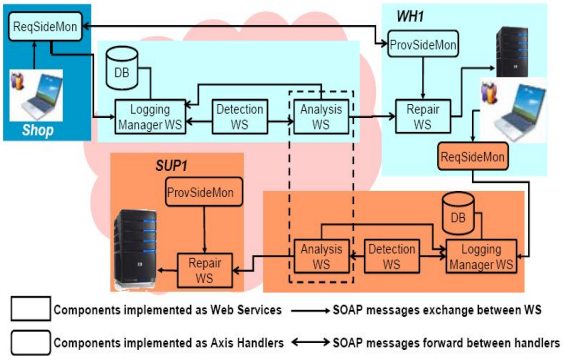

Fig 1. Details of QoS Architectural Framework Applied to the Food Shop Scenario

Figure 2, shows the overall monitoring result of the Web service using a self-healing proposed framework by Ben Halima et al. [4], The Figure shows that both curves are similar and the overload of monitors is negligible for less than 50 concurrent clients. For the largest requester's number (500) curve shows monitoring overload and its effect on the response time of web services.

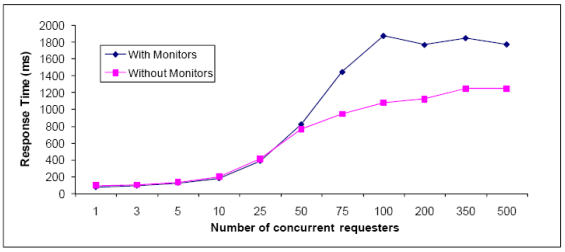

Fig 2. Overload of Monitoring.

II. QUALITY OF SERVICE DRIVEN

Web services' QoS is the factor for the success of service provides, which is related to Web services' functional and non-functional requirements; Verifying that the expected behavior has been delivered

There are number of Web service QoS characteristics that can be considered [5, 6, 7, 8, 15, 19, 20], shown in table 1.

Table 1: Web Service QoS Characteristics

Parameter	Description
Performance	A set of factors defining the productivity of a service: throughput, latency; - Throughput: number of requests that can be addressed in a given time-frame. - Latency: round-trip time of a request and its response.
Capacity	The limit of concurrent requests for guaranteed performance.
Execution cost	Cost The amount of money for a single service execution.
Compensation rate	A percentage of the execution price that is refunded when the service provider cannot deliver the ordered commodity.
Penalty	A percentage of the original price service requesters need to pay to the provider When he/she wants to cancel the committed service or the ordered item.
Availability	A set of metrics that characterize availability of a service: availability rate, mean time to repair, mean time between failures; how long a given service remains unavailable after occurrence of a failure.
Accessibility	Ability of a service to process a given request. The service can be available but not accessible if, for example, the hosting server is overloaded.
Network factors	A set of factors characterizing a communication network: network delay, delay Variation, packet loss.
Reliability	Reflects the ability of a service to keep operating over time, characterized by availability/accessibility and successful execution rate, and guarantee the promised or negotiated qualities of service, i.e. a percentage of successful executions with respect to all executions. according to Mani and Nagarajan [22] "*Reliability is the quality aspect of a Web service that represents the degree of being capable of maintaining the service and service quality.[...] reliability refers to the assured and ordered delivery for messages being sent and received by service requestors and service*

ISSN 2319-7595

International Journal of Information Systems and Computer Sciences, Vol.7. No.1, Pages : 16-22 (2017)
Special Issue of ICSIC 2017 - Held during 23-24 September 2017 in Amman Arab University, Amman-Jordan
http://www.warse.org/IJISCS/static/pdf/Issue/icsic2017sp16.pdf

	providers."
Scalability	The ability to process more operations in a given period.
Accuracy	The degree of conformity of the result produced by a service to an accepted standard value, or ideal value.
Exception handling	Specifies how a Web service handles exceptions.
Robustness	Reflects the ability of a service to function in the presence of invalid, incomplete or conflicting inputs.
Completeness	The discrepancy between specified and implemented features.
Security	A set of factors reflecting the trustworthiness of the service; Used to Guarantee exchanged messages confidentiality, non-repudiation, and encryption.
Privacy	Refer to the control of the personal data of a requester (ex. storage period, cases of disclosure).
Reputation	The average rate of service reported by clients can be considered as a trust measure to this service.

III. PROPOSED ARCHITECTURE

Our proposed architecture provides the Self-Healing capabilities base on three levels; Service, Flow, and communications levels. The following are the performance evaluation parameters for the proposed architecture using the QoS driven parameters in presence of the three levels Self-Healing:
The QoS parameters for the Service level (Web Service with Self-Healing Agent) are:

1- Execution Price: the fee that the Web service requester has to pay for invoking a Web service operation. it can be inquired by the requester or advertised by service providers; Ep(ws, op)
   - Ep: Web service's execution price
     - ws: invoked Web service
     - op: invoked Web service operation

2- Execution Time: the time in seconds between the moment when the request is sent by requester to a Web service and when the result of the request is received; Et(ws, op) + Tt(ws, Req) + Tt(ws, Resp) , calculated using Equation 1.
   - Et(ws, op): execution time needed by the Web service to process an operation.
   - Tt(ws, Req): transmission time needed for the request to reach the Web service.
   - Tt(ws, Resp): transmission time needed by the Web service response to reach the requester.

3- Internal Execution Time: the time in seconds between the moment when the request is received by a Web service and when the result of the request is sent; Et(ws, op), calculated using Equation 2.
   - Et(ws, op): execution time needed by the Web service to process an operation.

4- Reliability: the ability of a Web service to keep operating over time; measured by the number of failures per month/Year ;Rel(ws) = F(ws)/t
   - Rel(ws): Web service's reliability
   - F(ws): number of Web service's failures
   - t: period of time day, month, year, etc…)

5- Throughput: the amount of requests that can be processed by a Web service in a specific period of time; Thr(ws) = N(ws)/t.
   - Thr(ws): throughput of the Web service
   - N(ws): number of Web service's successful invocations
   - t: specific period of time (day, month, year, etc…)

6- Availability: the probability that a Web service is available and ready to use;
Av(ws) = 1 – ((Rej(ws, Req) + F(ws)) / K)
   - Av(ws): Web service's availability
   - Rej(ws, Req): Web service's rejected requests
   - F(ws): Number of Web service's failures
   - K: total number of requests

7- Response Time (Connector - To - Web Service): the time in seconds between the moment when the request is sent by the Virtual Web Service Connector to the Web Service, and when the result of the request is received by the Virtual Web Service Connector. Calculated using Equation (7).

8- Communication Time (Connector - To - Web Service): the round trip of a request and its response; the time in seconds between the moment when the request is sent by the Virtual Web Service Connector and Received by the Web service, and between the moment when the response is sent by the Web service received by the

ISSN 2319-7595

International Journal of Information Systems and Computer Sciences, Vol.7. No.1, Pages : 16-22 (2017)
Special Issue of ICSIC 2017 - Held during 23-24 September 2017 in Amman Arab University, Amman-Jordan
http://www.warse.org/IJISCS/static/pdf/Issue/icsic2017sp16.pdf

Virtual Web Service Connector. Calculated using Equation (6).

9- Accessibility: the ability of a Web service to process a given request; Acc(ws) = P(ws)/K , it depends on the Input Fault (IF), and the Input Type Fault (ITF).
   - Acc(ws): accessibility of the Web service
   - P(ws): number of requests processed by Web Service
   - K: total number of requests
10- Integrity: how the Web service maintains the correctness of the transactions' execution; provides the expected behavior; Int(ws) = Sc(ws,op) / Ex(ws).
   - Int(ws): integrity of the Web service
   - Sc(ws): number of successful Web service's operations execution.
   - Ex(ws): Total number of Web service operations execution.
11- Performance: the Web service's throughput and latency, it defines the productivity of the Web service; Throughput is the number of requests that can be addressed in a given time-frame. Latency is the round-trip time of a request and its response.

While The QoS parameters for the Communication level (Virtual Web Service Connector) are:

1- Execution Time: the time in seconds between the moment when request is sent by requester to the Virtual Web Service Connector, and receiving the response of the request. Calculated using Equations (3), (4) and (5).
   Execution Time = Eq(3) + Eq(4) + Eq(5)
2- Reliability: the ability of the Virtual Web Service Connector to keep operating over time; measured by the number of failures per month/Year; Rel(con) = F(con)/t.
   - Rel(con) : Virtual Web Service Connector's reliability
   - F(con): number of Virtual Web Service Connector's failures.
   - t: period of time day, month, year, etc…)
3- Throughput: the amount of requests that can be processed by the Virtual Web Service Connector in a specific period of time; Thr(con) = N(con)/t.
   - Thr(con): throughput of the Virtual Web Service Connector.
   - N(con): number of the Virtual Web Service Connector's successful invocations
   - t: specific period of time (day, month, year, etc…)
4- Availability: the probability that the Virtual Web Service Connector is available and ready to use; Av(con) = 1 – ((Rej(con, Req) + F(con))/K).
   - Av(con): Virtual Web Service Connector availability
   - Rej(con, Req): Virtual Web Service Connector's rejected requests
   - F(con): number of Virtual Web Service Connector's failures
   - K: total number of requests
5- Communication Time: the round trip of a request and its response; the time in seconds between the moment when the request is sent by the requester and received by the Virtual Web Service Connector, and between the moment when the response is sent by Virtual Web Service Connector and received by the requester. Calculated using Equation (8).
6- Response Time (Requester - To – Virtual Web Service Connector): the time in seconds between the moment when the request is sent by the requester to the Virtual Web Service Connector, and when the result of the request is received by the requester; calculated using Equation (1).
7- Accessibility: the ability of Virtual Web Service Connector to process a given request; Acc(con) = P(con)/K , it depends on the Input Fault (IF), and the Input Type Fault (ITF).
   - Acc(con): accessibility of the Virtual Web Service Connector
   - P(con): number of requests processed by the Virtual Web Service Connector.
   - K: total number of requests
8- Integrity: how the Virtual Web Service Connector maintains the correctness of the transactions' execution; provides the expected behavior; Int(con) = Sc(con,op)/Ex(con).
   - Int(con): integrity of the Virtual Web Service Connector
   - Sc(con, op): number of successful Virtual Web Service Connector's monitoring operations' execution.
   - Ex(con): total number of Virtual Web Service Connector monitoring operations execution.
9- Performance: Virtual Web Service Connector's throughput and latency, it defines the productivity of the Virtual Web Service Connector; Throughput is the number of requests that can be addressed in a given time-frame. Latency is the round-trip time of a request and its response.
10- Internal Monitoring Time: the time in seconds of the monitoring operations used to

ISSN 2319-7595

International Journal of Information Systems and Computer Sciences, Vol.7. No.1, Pages : 16-22 (2017)
Special Issue of ICSIC 2017 - Held during 23-24 September 2017 in Amman Arab University, Amman-Jordan
http://www.warse.org/IJISCS/static/pdf/Issue/icsic2017sp16.pdf

add the QoS variables to the request's SOAP message, and to extract in the results from the response's SOAP message. Calculated using Equations (3) and (4); Total monitoring time = Eq(3) + Eq(4).

Finally, The QoS parameters for the Flow level (Extended Execution Engine with Process Execution Self-Healing Manager) are:

1- Reliability: ability of a Extended Execution Engine with Process Execution Self-Healing Manager to keep operating over time; the ability of the composed Web service to keep operating over time, measured by the number of failures per month/Year; Rel(EEeng) = F(EEeng)/t.
   - Rel(EEeng) : Extended Execution Engine with Process Execution Self-Healing Manager's reliability
   - F(EEeng): number of Extended Execution Engine with Process Execution Self-Healing Manager's failures; unsuccessful invocations' order according to the business process.
   - t: specific period of time day, month, year, etc…)
2- Throughput: the amount of business process that can be processed by Extended Execution Engine with Process Execution Self-Healing Manager in a specific period of time; Thr(EEeng) = N(EEeng)/t.
   - Thr(EEeng): throughput of Extended Execution Engine with Process Execution Self-Healing Manager.
   - N(EEeng): number of business processes processed by the Extended Execution Engine with Process Execution Self-Healing Manager.
   - t: specific period of time day, month, year, etc…)
3- Availability: the probability that the Extended Execution Engine with Process Execution Self-Healing Manager is available and ready to use; Av(EEeng) = 1 – ((Rej(EEeng, Notifications) + F(EEeng))/K)
   - Av(EEeng): Extended Execution Engine with Process Execution Self-Healing Manager availability
   - Rej(EEeng, Notifications): Extended Execution Engine with Process Execution Self-Healing Manager's rejected notifications to execute a business process.
   - F(EEeng): Number of Extended Execution Engine with Process Execution Self-Healing Manager's failures
   - K: total number of notifications
4- Accessibility: ability of Extended Execution Engine with Process Execution Self-Healing Manager to process a given business process; Acc(EEeng) = P(EEeng) / K , depends on the Order Fault (OF).
   - Acc(EEeng): Extended Execution Engine with Process Execution Self-Healing Manager Accessibility
   - P(EEeng): number of business process processed by Extended Execution Engine with Process Execution Self-Healing Manager
   - K: total number of business processes notified to be executed by Extended Execution Engine with Process Execution Self-Healing Manager
5- Integrity: how Extended Execution Engine with Process Execution Self-Healing Manager maintains the correctness of the transactions' execution; provides the expected behavior of the Extended Execution Engine with Process Execution Self-Healing Manager; Int(EEeng) = Sc(EEeng,BP) /Ex(EEeng).
   - Int(EEeng): integrity of the Extended Execution Engine with Process Execution Self-Healing Manager
   - Sc(EEeng, BP): number of successful Extended Execution Engine with Process Execution Self-Healing Manager's business processes execution.
   - Ex(EEeng): Total number of Extended Execution Engine with Process Execution Self-Healing Manager's business processes execution.
6- Performance: Extended Execution Engine with Process Execution Self-Healing Manager's throughput and latency, it defines the productivity of the Extended Execution Engine with Process Execution Self-Healing Manager; Throughput is the number of business processes that can be addressed in a given time-frame. Latency is the round-trip time of a business processes invocation and its response.
7- Execution price: The execution price of an execution plan of a composite service is the sum of the execution prices of the operations invoked over the services that participate in composite service.
8- Execution duration: The execution duration of an execution plan of a composite service is the sum of the execution time of the Web services that participates in composite service; the time in seconds between the

ISSN 2319-7595

International Journal of Information Systems and Computer Sciences, Vol.7. No.1, Pages : 16-22 (2017)
Special Issue of ICSIC 2017 - Held during 23-24 September 2017 in Amman Arab University, Amman-Jordan
http://www.warse.org/IJISCS/static/pdf/Issue/icsic2017sp16.pdf

moment when the business process first request is sent by the Extended Execution Engine with Process Execution Self-Healing Manager to the participating Web service, and when the result of the business process's last request is received by the Extended Execution Engine with Process Execution Self-Healing Manager.

In our proposal architecture, QoS parameters are measured while considering the following eight values, shown in Figure 4. t1: the time at which Service Requester (Extended Execution Engine with Process Execution Self-Healing Manager) has issued the request. t2: the time at which Virtual Web Service Connector has received the request. t3: the time at which Virtual Web Service Connector has issued the request. t4: the time at which Service Provider has received the request. t5: the time at which Service Provider has issued the response. t6: the time at which Virtual Web Service Connector has received the response. t7: the time at which Virtual Web Service Connector has issued the response. t8: the time at which Service Requester (Extended Execution Engine with Process Execution Self-Healing Manager) has received the response.

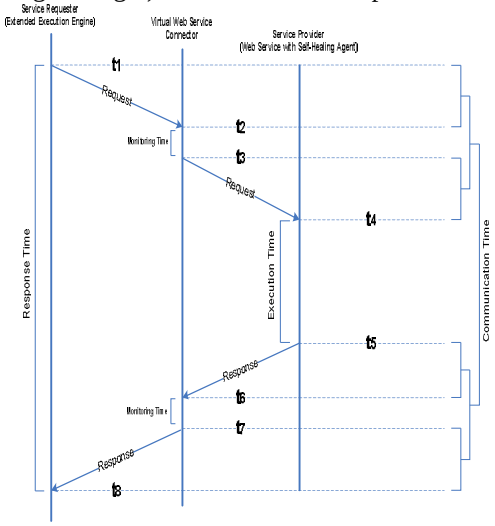

Fig. 4 Proposed architecture measured times for QoS Monitoring.

The QoS parameters is will be calculated using the following equations:

Response time = t8 – t1 (1)

Execution time = t5 – t4 (2)

Virtual W-Service Connector Req. monitoring time = t3 – t2 (3)

Virtual Web Service Connector Res. monitoring time = t7 – t6 (4)

Total Communication time = (t2 – t1) + (t4 – t3) + (t6 – t5) + (t8 – t7) (5)

Connector – Web Service Communication time = (t4 – t3) + (t6 – t5) (6)

Web Service Response Time = t6 – t3 (7)

Requester – Conn-Comm-time = (t2 – t1) + (t8 – t7) (8)

IV. CONCLUSION

An exact differentiation between Service oriented architecture and component based architecture is hard to make, because opinions on what "SOA" exactly is and how it will develop differ. If SOA is seen as a new type of architecture that defines the how-to of assigning interfaces in a servicing way so that these services can be used in a context freeway, it does not differ significantly from existing component based frameworks like Enterprise JavaBeans. If the definition of SOA includes the usage of technologies like WSDL, UDDI, and SOAP (and its potential successors), SOA differs in several ways from the "old" component based architecture. With these technologies, software can be built in a completely new way. Software developers can use foreign, external "components" in the form of Web Services. Web Services can be used in contexts that were not considered at the time they were built. But SOA is not the solution to all problems linked with software development. There are many problems: Ranging from finding the required services, providing acceptable performance, security, realising transactions up to maintaining one's own service, even if foreign, integrated services have changed or are closed. There are many problems to resolve, but there are many possibilities too. It will depend on Sun or other larger companies, to develop an overall solution, containing solutions to all of these problems.

REFERENCES

[1] Rajeev Alur and David L. Dill. A theory of timed automata. Theor. Comput. Sci., 126(2):183–235, 1994.

[2] Natee Artaiam and Twittie Senivongse. Enhancing service-side qos monitoring for web services. In SNPD '08: Proceedings of the 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pages 765–770, Washington, DC, USA, 2008. IEEE Computer Society.

[3] Luciano Baresi, Carlo Ghezzi, and Sam Guinea. Smart monitors for composed services. In ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing, pages 193–202, New York, NY, USA, 2004. ACM.

[4] Bruce Bukovics. Pro WF: Windows Workflow in .NET 3.5. Apress, Berkely, CA, USA, 2008.

[5] Michael J. Carey. Soa what? Computer, 41(3):92–94, 2008.

[6] Issam Chebbi, Schahram Dustdar, and Samir Tata. The view-based approach to dynamic inter-organizational workflow cooperation. Data Knowl. Eng., 56(2):139–173, 2006.

[7] OASIS International Standards Consortium. Uddi version 3.0.2, September 2004. http://www.uddi.org/pubs/uddi v3.htm.

[8] OASIS International Standards Consortium. ebxml registry services and protocols v3.0, March 2005. http://www.ebxml.org.

[9] Thomas Erl. Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[10] Al_Zyadat, W., Al-Zyoud, F., and Alhroob , A.,Smooth Handoff Process Cluster_Based In Vehicular Ad Hoc Networks, International Journal of Computing Academic Research (IJCAR), Vol. 6(3), 101-109, 2017.

[11] Organization for the Advance of Structured Information Standards (OASIS). Reference model for service oriented architecture 1.0, oasis standard, October 2006. http://docs.oasis-open.org/soarm/v1.0/.

[12] Khalid A. Kaabneh, "High Fidelity Image Watermarking Using FWT by Exploiting the RGB and HSV Models", International Journal of Computer Technology and Applications (IJCTA), V. 5(2), April 2014.

[13] Al-Hadid, I. , Airport Enterprise Service Bus with Self-Healing Architecture (AESB-SH), International Journal of Aviation Technology, Engineering and Management, Vol. 1(1), 1-13, 201.

[14] Lican Huang, D.W. Walker, O.F. Rana, and Yan Huang. Dynamicworkflow management using performance data. In Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on, volume 1, pages 154–157, May 2006.

[15] Afaneh, S., Al-Hadid, I., Al-Malahmeh, H., "Extreme Programming Agile Methodology With Self-Healing", In Proceedings of the European, Mediterranean and Middle Eastern Conference on Information Systems (EMCIS), Germany. 2012.

[16] Philipp Leitner, Florian Rosenberg, and Schahram Dustdar. Daios: Efficient dynamic web service invocation. IEEE Internet Computing, 13(3):72–80, 2009.

[17] Yousef, Nidal, Hassan Altarwaneh, and Aysh Alhroob. "Best Test Cases Selection Approach Using Genetic Algorithm." Computer and Information Science 8.1, 2015.

[18] Al-Hadid, I., Khwaldeh, S., Kaabneh, k., Tarawneh, H.,, Efficient Big Data Transfer Technique for Static Routing Networks, International Journal of Applied Engineering Research, Vol 12 (9), pp 2071-2078, 2017.

[19] Afaneh, S., Al-Hdid, I., Airport Enterprise Service Bus with Three Levels Self-Healing Architecture (AESB-3LSH), International Journal of Space Technology Management and Innovation (IJSTMI), Vol. 3 (2), 1-23, 2013.

[20] Al-hadid, I., Afaneh, S., and Almalahmeh, H., Extreme Programming Certification Process using Confidence Grade, International Journal of Computer and Information Technology, Vol. 4(1), pp 129 - 138, 2015.

[21] Kaabneh, K., Afaneh, S., Almalahmeh, H., Al-hadid, I. , Improved Web Service Self-Healing Connector, International Journal of Computer Science and Information Technologies, Vol. 5 (2), 2649 - 2657, 2014.

[22] Mani, A., and Nagarajan, A., "Understanding quality of service for Web services: Improving the performance of your Web services", Retrieved August 22, 2009 from http://www-106.ibm.com/developerworks/library/ws-quality.html