International Journal of Grid and Distributed Computing
Vol. 11, No. 3 (2018), pp.89-98
http://dx.doi.org/10.14257/ijgdc.2018.11.3.08

Middleware-based Model for Dynamic Reconfiguration of Web Service

Rahmat Ilahi, Novia Admodisastro, Norhayati Mohd Ali, Abu Bakar Md. Sultan

Department of Software Engineering and Information System
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia, Serdang, Malaysia
gs36028@student.upm.edu.my, {novia,hayati,abakar}@upm.edu.my

Abstract

Dynamic reconfiguration of web services (WSs) is a method to replace WSs in service-oriented architecture (SOA) software system during runtime. The method allows the service requester to replace WSs in their system without interruption of other operations in the system. However, dynamic reconfiguration is a difficult process and attempts to handle the process appropriately are still lacking. In this paper, we propose a middleware-based model to improve SOA dynamic reconfiguration service process during runtime. The model is used as the main standard to outline the dynamic reconfiguration of WSs that can handle quality of service (QoS) requirements and to provide explicit mechanism during pre-, in-, and post-adaptation stages. A self-adaptive tool was developed based on the model to support the dynamic reconfiguration of WSs with minimum human intervention. Finally, an evaluation using an experiment is conducted to evaluate the effectiveness of the model. The evaluation results show that the model supported the dynamic reconfiguration of WSs effectively.

Keywords*: dynamic reconfiguration, middleware, web services, SOA*

1. Introduction

In service-oriented architecture (SOA), the notion of services made possible by web services (WSs), which initially introduced by Jini technology [1]. In developing Service-Based System (SBS) that composed by a set of WSs, the service requester as a consumer of WSs may obtain WSs from either internally or third party. Consequently, the capabilities and qualities of SBS will more depend on the quality of the WSs. In dynamic of SOA, the SBS must be able to react to both user and environment changes during static or operational settings. Thus, an adaptation process is performed either by reconfigure the system and then restart it, or reconfigure the system during operation. A static adaptation is about performing the reconfiguration when the system can be shut down in order to make the required changes. A dynamic adaptation, also known as dynamic reconfiguration [2] is about performing the reconfiguration without stopping or restarting the system [3].

Several works that focus on dynamic reconfiguration are mainly aimed at the service composition level and using middleware-based approach [4-9]. However, these works only focus on certain stages of the dynamic reconfiguration process. Thus, in this paper, we proposed a middleware-based model called Dynamic REconfiguration of WS (DREWS) that intended to handle dynamic reconfiguration service during runtime. The model handled functional as well as the quality of service (QoS) requirements during dynamic reconfiguration service, and to provide explicit mechanism during pre-, in-, and post-adaptation stages. We then developed

Received (October 28, 2018), Review Result (December 18, 2018), Accepted (December 24, 2018)

a self-adaptive tool based on the model that supports minimum human intervention during the dynamic reconfiguration process.

This paper is organised as follows: Section 2 provides some background of WS dynamic reconfiguration in SOA; Section 3 described the model and its implementation tool; Section 4 presents the evaluation of DREWS by using experiment approach and a scenario, and finally, Section 5 provides concluding thoughts.

2. Background

Valls *et al.*, [4] described that a web-based system needs to modify their structure in runtime due to changes that occur in an expected, i.e. upgrading the version of system or unexpected situation, i.e. system failure. In SBS where it is composed of a collection of services, the possibility of system changes during runtime is higher because of several reasons, such as the unavailability of services [5,10]. As a result, the modification is inevitable and dynamic reconfiguration is required to handle service replacement. Krishnamurthy & Babu [7] categorised the need of dynamic reconfiguration into two main situations: upgrading WS version and WS failure. From SOA lifecycle perspective, a dynamic reconfiguration service process can be categorized into three stages:

a. *Pre-adaptation stage.* The stage refers to the preliminary stage before service reconfiguration is started. The main process involves in this stage is a selection of WS among a set of WS candidates that offers the same functionalities.

b. *In-adaptation stage.* The stage refers to where the actual service reconfiguration during runtime is started. The main process here is service reconfiguration and other important features are required, such as handling blackout and error exception, *etc.*

c. *Post-adaptation stage.* The stage refers to the preceding stage after service reconfiguration is completed. The main process here is to ensure the new adapted WS is able to be released to the real environment. Thus, important features are required, such as rollback and restoration management, etc.

3. The Model

In this section, we discuss DREWS, a middleware-based model to support dynamic reconfiguration of WSs during runtime. DREWS consists of three main processes: (i) Manage Adaptation Process (MAP), (ii) Selection Process (SP) and (iii) Reconfiguration Process (RP). The three processes are supported by the Connection and Log Recorder (CLR), a repository that holds reconfiguration data (Figure 1).

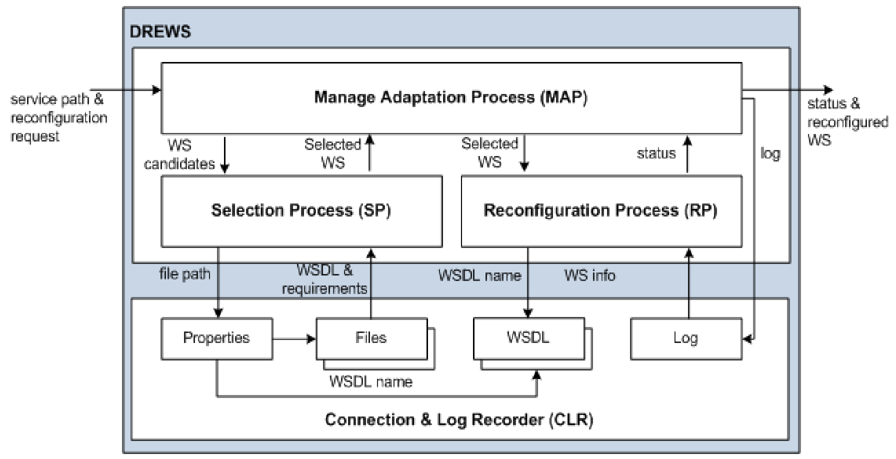

Figure 1. DREWS Model

DREWS underlie the dynamic reconfiguration process with the following tasks: WS selection, WS replacement, and WS verification. WS selection is a task in the pre-adaptation stage to validate a set of WS candidates which provides similar functionalities and to find the best WS among the WS candidates. The functional and QoS aspects are primary concerns to further constrain and select the best WS for a valid service reconfiguration. WS replacement is a task in the in-adaptation stage to reconfigure the existing WS by replacing and rerouting the WS with the WS chosen during the WS selection. The chosen WS is either provided by the same service provider of the previous WS or by different service providers. Finally, WS verification is a task in the post-adaptation stage to verify the proper binding of replacement WS to SBS. In order to conduct all of the processes, DREWS has to support two main attributes: functional attributes and QoS attributes.

The details of DREWS components are described below:

a. The MAP has two roles, first is to act as task manager to distribute tasks between SP and RP. Second, to verify reconfiguration status after the complete configuration of WS. The MAP is interacting with SP and RP processes with four steps as follow:

    i. *Receive adaptation request*. The SBS sends a reconfiguration request to the MAP to conduct dynamic reconfiguration service.

    ii. *Receive validation feedback*. The step is performed after receiving a validation feedback from the SP that indicates a WS has been selected. The feedback contains WS Definition Language (WSDL) URL of the selected WS.

    iii. *Verify reconfiguration status*. The MAP receives a reconfiguration status from RP which indicates the status of success or failure. Based on the status, the selected WS connectivity checks to ensure that it is connected properly to the SBS.

    iv. *Release WS*. MAP release system blocking and sent reconfiguration result status to the SBS. In a situation of adaptation request is upgrading service and reconfiguration result is failed, DREWS going to rollback to the previous WS connection.

b. The SP is a process to conduct web service selection for finding and validating suitable WS for replacement. The new WS is selected from a set of WS candidates registered in the CLR. Dynamic service environments cause some difficulties in service selection [11]. Two important factors are considered, *i.e.*, functional requirement and QoS to find the suitable new WS [12]. The SP is conducted at the pre-adaptation stage of dynamic reconfiguration process with four main steps:

i. *Get WS candidates*. When receiving requests from the MAP, SP starts to get the WS candidates information from the CLR.

ii. *Get WS requirements*. The reconfiguration requirements are retrieved from the CLR. The requirements that consist of functional and QoS aspects are determined by the service requester.

iii. *Compare WS candidates and WS requirements*. WS candidates that represented by WSDL file are compared with WS requirements. There are two types of requirements are compared: functional requirements and QoS.

iv. *Choose WS replacement*. The step is to deliver a most suitable WS based on the scoring values of the WS candidates. The highest score WS is sent to the MAP as the SP end result.

c. The RP main purpose is to conduct service reconfiguration during runtime. The process occurs during in-adaptation stage which requires a support from CLR. Before conducting reconfiguration process, RP will block incoming request to prevent operation failure when the reconfiguration process is conducted and backup existing configuration as error handling when reconfiguration failure occurred. RP consists of four main steps:

i. *Get WS connection info*. The MAP passes an input to the RP that contains the URL of the selected WS. The URL enables the RP to retrieve the selected WS WSDL file.

ii. *Block incoming process*. During reconfiguration, incoming requests to invoke the existing WS are put on hold. This step is executed to prevent operation failure during the reconfiguration process.

iii. *Backup existing connection*. In this step, the existing WS connection is copied, backed-up and stored into CLR.

iv. *Update configuration file.* The final step is to replace the existing connection information that resides in the CLR by information that was collected from the WSDL file of the new WS.

d. The CLR is a repository that used by DREWS to retrieve and record the WS information for reconfiguration service purpose. The CLR is located separately from DREWS to ensure the service requesters can manage the CLR dynamically without affecting DREWS main structure. There are five main functions of the CLR:

i. *Storing WS path file.* This file is a parent file where it is used by DREWS to call all other files stored in CLR.

ii. *Storing WS* configuration *information.* WS configuration information resides in a *serviceconfig* file with the aims to minimise connection dependency between SBS and WS.

iii. *Storing WS requirements*. WS requirements that consist of functional requirement and QoS is recorded in CLR.

iv. *Storing WS candidates*. The CLR is used to store the WS candidates. When SP is started the WS candidate file going to be used to access URL of the WS candidates.

v. *Logging reconfiguration activities*. The entire process of service reconfiguration is stored in a log file.

3.1. DREWS Attributes

DREWS has to support two main attributes to perform dynamic reconfiguration of WS as described below:

a. Functionalities Attributes. WSDL file contains information about WS parameter, data connectivity, binding, functionalities of WS, and message exchange protocol [13]. It acts as an interface to invoke WS from the SBS. The main purpose of WSDL is to support DREWS SP and RP processes. In SP, WSDL file is used to validate WS functionalities, *i.e.*, WS can calculate the shipping price. While in RP, WSDL file is used to get the WS functionalities and connectivity. WS functionalities are represented by <*operation*> tag in WSDL file.

b. QoS Attributes. The WSDL file describes all information related to WS functionalities, connectivity and messages exchange but does not contain any information related to QoS. QoS attributes are a crucial part of WS where it determines user satisfaction when using the WS [14]. Therefore, DREWS has includes the QoS attributes by extending WSDL file to include QoS descriptions. DREWS supports four main QoS attributes that commonly used by service requester to find suitable WS [15] as described below:

   i. *Service reputation*. The reputation of WS is evaluated by service requesters that used the WS previously.

   ii. *Availability*. The attribute is to ensure the WS is available in their location or the WS is available when required.

   iii. *Response time*. Service requester must ensure that the new WS response time is better or at least same with the existing WS.

   iv. *Throughput*. When selecting a new WS, one of consideration is the WS must able to receive many requests for its operation at the same time without prejudice the WS performance.

3.2. The DREWS Tool Support

In this section, the tool support underlying the DREWS model is discussed. The tool is developed using J2EE and Apache CXF [16]. DREWS tool consists of four main components which provide feature as dynamic reconfiguration service executor and supported by a file repository as shown in Figure 2:

a. *Adaptation Manager (AM)*. A component to interact with SBS and WS. This component distributes, manages and monitors the overall dynamic reconfiguration process.

b. *Service Selection Agent (SSA)*. A component to find and validate a new most suitable WS to be adapted.

c. *Service Reconfiguration Agent (SRA)*. A component to conduct reconfiguration service during runtime by replacing existing WS with the new selected WS from SSA.

d. *File Repository*. A repository to store several different files and specifications that include *WS* path properties, WS candidates, WS requirements, service configuration properties and log file.

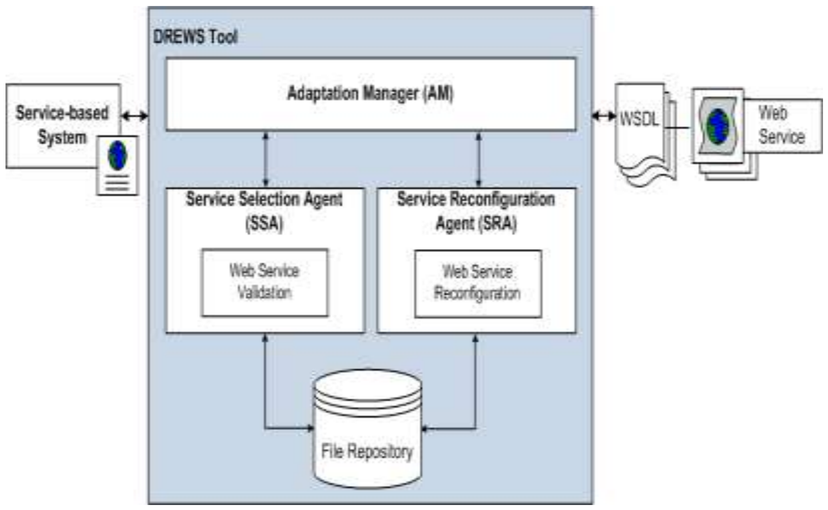

Figure 2 DREWS Tool Architecture

4. Evaluation

An experiment has been conducted to evaluate the effectiveness of DREWS in handling dynamic reconfiguration during pre- (*i.e.*, find WS, identify functionalities, identify QoS and calculate QoS), in- (*i.e.*, conduct reconfiguration) and post- (*i.e.*, test connectivity and release WS) adaptation by comparing between DREWS and Manual Reconfiguration (MR) of WS with two main objectives. First, to observe the correctness and difficulty of both methods to conduct dynamic reconfiguration service. Second, to compare their reconfiguration time. The manual approach is a way to evaluate the system by using test case that has prepared manually and execute them to identify defects in the system [17]. Ten peoples who working and experience in software development are invited to become participants as shown in Table 1. In the MR approach, the participants are required to conduct similar processes that DREWS conducted but using human intervention.

Table 1. Demography of Participants

Number of software developers	Experience in system development	Experience in SOA development	Experience in WS integration and configuration
5	1-2	2	0
5	3-5	0	0

4.1. The Scenario

Courier Online System (COS) has been developed based on SOA to evaluate DREWS. COS is a system that handles courier shipment daily operations from ordering shipment, checking shipment cost and tracking shipment delivery with developed its modules by using WS. Figure 3 shows the COS system architecture that choreographs and coordinates three different services into a workflow to establish the business processes. In addition, the COS considers relevant QoS aspects in delivering these services to their customers. Thus, a set of QoS requirements is specified and stored in the repository. The COS uses Apache CFX to interact with the DREWS tool that separates the COS with WS reconfiguration settings. Thus, it allows the COS submits independent reconfiguration requests to DREWS tool during system runtime.

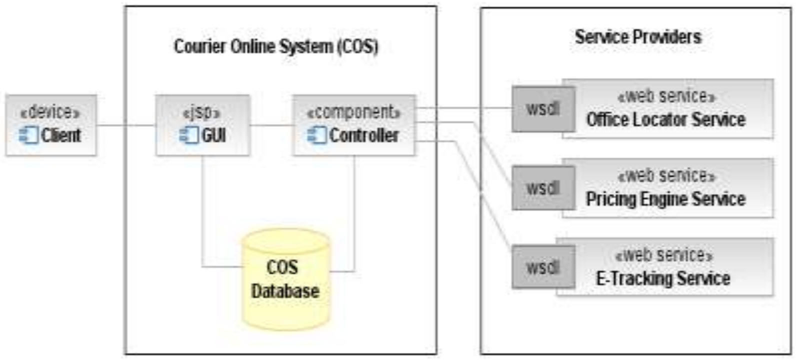
Courier Online System (COS)
«device» Client
«jsp» GUI
«component» Controller
COS Database
Service Providers
wsdl
«web service» Office Locator Service
«web service» Pricing Engine Service
«web service» E-Tracking Service

Figure 3. The COS System Architecture

4.2. Result and Discussion

The first experiment was executed by using the MR approach. The results showed that all participants could execute all tasks completely and correctly, but with different levels of difficulty as shown in Figure 4. The results show the most difficult task was to conduct pre-adaptation stage where they need to identify WS functionalities and QoS, calculate QoS value its total score. The accuracy of calculation depends on the participants understanding and skill based on the formulas given. The difficulty level of the other tasks varies, but they agree that the process from beginning until the end take quite long and the operator need to understand the whole reconfiguration process before they can conduct reconfiguration service.

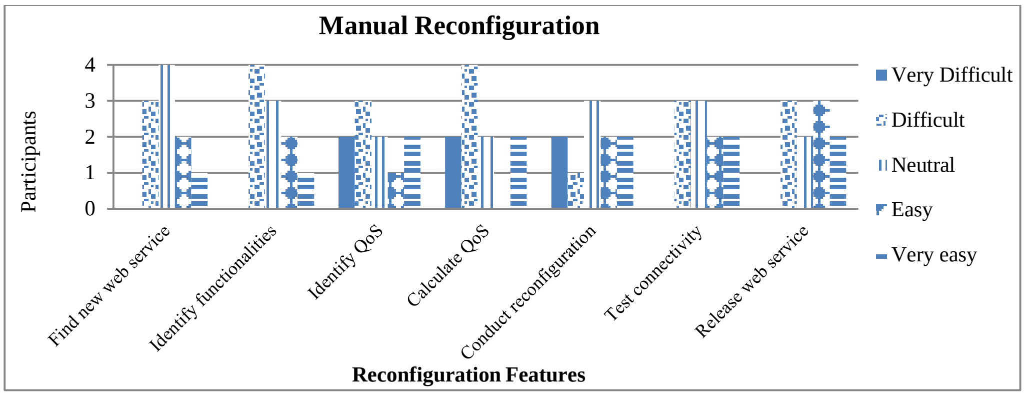

Figure 4. MR Service Execution Result

The next experiment is to conduct an experiment by executing the COS and triggering the DREWS tool. The results showed the DREWS tool could conduct the dynamic reconfiguration service automatically. The participants only needed to monitor the processes and they did not need to do complicated calculation and configuration. As a result, all participants agreed the DREWS tool has abilities to accomplish all tasks and processes correctly and easily. As shown in Figure 5, all participants gave the highest score to all tasks execution that indicates DREWS able to conduct dynamic reconfiguration service effectively as compared to MR approach.

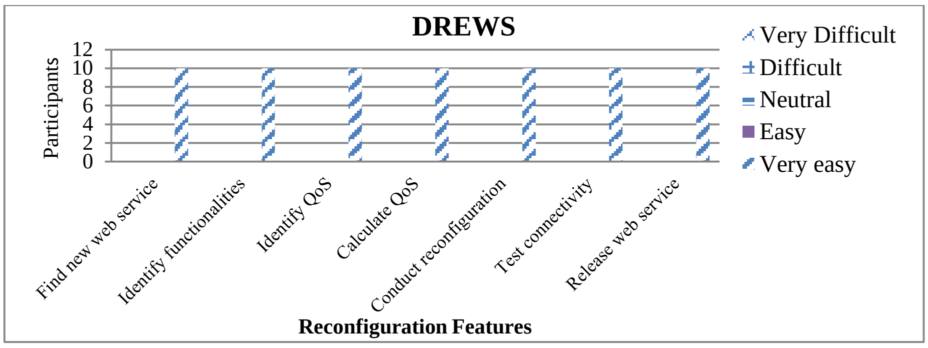

Figure 5. DREWS Tool Reconfiguration Service Result

The reconfiguration time is an important factor in WS reconfiguration because during reconfiguration, the system cannot receive any request to execute the specific operation due to unreachable WS [18]. After the participants executed both approaches, it showed that the reconfiguration service time, which conducted by DREWS tool significantly faster compared to manual approach as shown in Figure 6. The Table 2 shows that the average time that needed by the participants to conduct MR is 798.6 second. While, DREWS tool only needs average 0.62 seconds to conduct same processes.

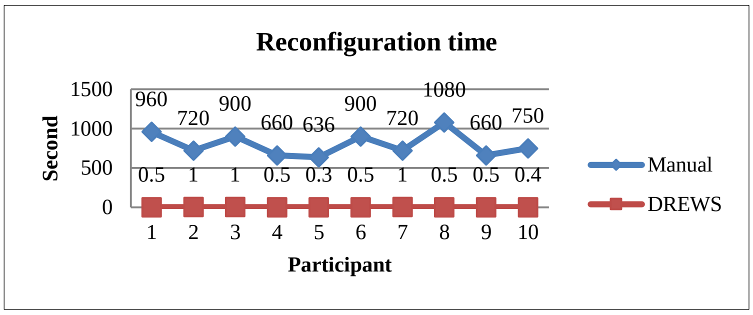

Figure 6. Reconfiguration Time – DREWS Tools vs MR

The evaluation also discovers the most consuming time in the MR process was to find suitable WS for replacement. As a conclusion, the comparison between MR service and the DREWS tool showed that the DREWS tool is able to conduct dynamic reconfiguration service effectively compared to MR method. Additionally, the DREWS also able to prevent any incoming request during the reconfiguration process that helps to reduce operation failure during the reconfiguration process. Besides, the DREWS is able to handle reconfiguration failure by using rollback mechanism and provide useful information for the user's references.

Table 2. Reconfiguration Time

Reconfiguration methods	Max. time (second)	Min. time (second)	Average time (seconds)
MR	1080	660	798.6
DREWS	1	0.3	0.62

5. Conclusion

This paper presents a middleware-based model called DREWS to handle dynamic reconfiguration of WS during runtime without turning off the system. The model handled dynamic reconfiguration service during pre-, in- and post-adaptation stages.

A tool based on the DREWS model is developed and illustrated in COS. For the evaluation purposed, the experiment has been conducted by comparing the DREWS with MR approach. The results show DREWS has abilities to conduct dynamic reconfiguration of WS effectively without human intervention.

Acknowledgement

The authors are grateful to Universiti Putra Malaysia and the Ministry of Higher Education, Malaysia Government via the FRGS grant for support of this research.

References

[1] Oracle, **(2016)**, http://www.oracle.com/technetwork/articles/javase/soa-142870.html. Date accessed: 04/06/2016.

[2] P. Grace, “Dynamic Adaptation”, Middleware for Net. Eccentric and Mobile Apps, **(2009)**, pp. 285-302. doi:10.1007/978-3-540-89707-1_13.

[3] A. D’Ambrogio, “Model-driven quality engineering of SBS”, G. A. Tsihrintzis, M. Virvou & L. C. Jain (Eds.), Multimedia services in intelligent environments: Software development challenges and solutions **(2010)**, pp. 81-103, doi:10.1007/978-3-642-13355-8_6.

[4] M. G. Valls, I. R. Lopez and L. F. Villar, ILAND: An Enhanced Middleware for Real-Time Reconfiguration of Service Oriented Dist. Real-Time Systems. IEEE Transactions on Industrial Informatics IEEE Trans. Ind. Inf., vol. 9, no. 1, **(2013)**, pp. 228-236. doi:10.1109/tii.2012.2198662.

[5] H. Gomaa, & K. Hashimoto, “Dynamic self-adaptation for dist. service-oriented transactions”, Software Engineering for Adaptive and Self-Managing Systems (SEAMS), **(2012)**, pp. 11-20. doi:10.1109/SEAMS.2012.6224386.

[6] V. Krishnamurthy and C. Babu, “Dynamically reconfiguring services in SOA apps.: A pattern-based approach”, Proc. of the 17th European Conference on Pattern Languages of Programs, Germany. , vol. 6, **(2012)**, pp. 1-13. doi:10.1145/2602928.2603082

[7] K. Lin, M. Panahi and Y. Zhang, “The Design of an Intelligent Accountability Architecture’, Int. Conf. on E-Business Engi. **(2007)**, doi:10.1109/icebe.2007.4402087.

[8] K. Lin, J. Zhang and Y. Zhai, “An Efficient Approach for Service Process Reconfiguration in SOA with End-to-End QoS Constraints”, “Conf. on Commerce and Enterprise Comp”, **(2009)**, doi:10.1109/cec.2009.87.

[9] K. Lin, J. Zhang, Y. Zhai and B. Xu, “The design and implementation of service process reconfiguration with end-to-end QoS constraints in SOA”, SOCA Service Oriented Comp. and Apps., vol. 4, no. 3, **(2010)**, 157-168. doi:10.1007/s11761-010-0063-6.

[10] J. L. Fiadeiro and A. Lopes, “A model for dynamic reconfiguration in SOA”, Softw Syst Model Software & Systems Modeling, vol. 12, no. 2, **(2012)**, pp. 349-367. doi:10.1007/s10270-012-0236-1.

[11] S. Maheswari and G. R. Karpagam,“Comparative Analysis of Semantic WS Selection Methods”, Indian Journal of Sc. and Tech., vol. 8, no. 3, **(2015)**, p. 159. doi:10.17485/ijst/2015/v8i1/60499

[12] Y. Gong, L. Huang, F. Jiang and K. Han, “An approach to WS dynamic replacement”, Int. J. of Grid and Dist. Comp., vol.7, no. 1, **(2014)**, pp.1-12. doi:10.14257/ijgdc.2014.7.1.01

[13] WS Description Language (WSDL) 1.1., **(2001)**, https://www.w3.org/TR/wsdl. Date accessed: 18/12/2013.

[14] Y. Li, X. Zhang, Y. Yin and J. Wu, “QoS-Driven Dynamic Reconfiguration of the SOA Based Software”, Int. Conf. on Service Sc. **(2010)**, doi:10.1109/icss.2010.58.

[15] M. Karimi, F. S. Esfahani, and N. Noorafza, “Improving Response Time of WS Composition based on QoS Properties”, Indian J. of Sc. and Tech., **(2015)**, vol. 8, no. 16. doi:10.17485/ijst/2015/v8i16/55122.

[16] Apache CXF, Apache CXF: An Open-Source Services Framework, **(2016)**. http://cxf.apache.org/. Date accessed: 10/03/2016.

[17] R. M. Sharma, “Quantitative analysis of automation and manual testing”, IJEIT, vol. 4, no. 1, **(2014)**, pp. 252-257.

[18] U. Brinkschulte, E. Schneider and F. Picioroaga, “Dynamic real-time reconfiguration in dist. Systems”, Timing issues and solutions, **(2005)**, ISORC. doi:10.1109/ISORC.2005.25.