

# A Dynamic Reconfiguration Model of Web Services in Service-Oriented Architecture

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

*Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology,  
Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia.  
gs36028@student.upm.edu.my*

**Abstract**—Service-Oriented Architecture (SOA) makes it possible to build distributed systems with web services that can be looked up, published and bound on the execution time across the boundary of an organisation over the Internet. By using standard interfaces and message-exchanging protocols, developers are able to reuse existing web services and integrate these individual services. Nevertheless, SOA must be able to provide a way to cope with dynamic changes that may occur in the system requirements and the environment in which the system operates. The means is known as dynamic reconfiguration that allows web services binding happens at runtime by matching the functional as well as Quality of Service (QoS) requirements to ensure dependable SOA systems. In the paper, we introduce a dynamic reconfiguration of web services model (DREWS) using middleware-based approach. The model intended to handle functional and QoS requirements during dynamic reconfiguration process and to provide an explicit mechanism during pre-, in-, and post-adaptation stages. A self-adaptive tool is developed based on the model to support the dynamic reconfiguration process that allows minimum human intervention.

**Index Terms**—Dynamic Reconfiguration; Middleware-based; Service-Oriented Architecture; Web Service.

## I. INTRODUCTION

Service-Oriented Architecture (SOA) enables the development of flexible, efficient and evolving distributed service-based systems [1]. SOA is defined as an approach to develop service-based systems by integrating independent Web Service (WS) [2, 3]. The WSs communications involve simple data passing, or involve two or more WSs coordinating some activities. Building the service-based systems face an open and heterogeneous computing environment, where numerous distributed WSs over the Internet perform computation concurrently and collaboratively. While, SOA concerns on service standards, protocol standards, cross-enterprise application and the direct interaction between service requestor and service provider, the dynamic nature of the business environment requires service-based systems to be highly reactive and adaptive [4].

Thus, a means to ensure the service-based systems capable to be adapted to meet changing requirements is crucial. It is also to ensure the systems could be adapted to the demands of rapidly changing environments. Due to the systems have to work in a large-scale open environment where the WSs are subject to constant changes and variations. The WSs evolve due to changes in structures, behaviour and policies. Despite the WSs volatility, developers have to ensure the Quality of Service (QoS) properties and to make intelligent use of new WSs. Such changes can be identified, detected, and foreseen

in the service-based systems during monitoring of the systems execution and its environment. In such a setting, adaptation is necessary to modify service-based systems so as to satisfy new requirements as dictated by the changes of the environment.

Therefore, we have developed a dynamic reconfiguration of WSs (DREWS) model using middleware-based approach to ensure dependable SOA systems during runtime. The model is intended to handle functional as well as QoS requirements during dynamic reconfiguration process, and to provide explicit mechanism during pre-, in-, and post-adaptation stages. Then, a self-adaptive tool is developed based on the model that allows less human intervention during dynamic reconfiguration process.

This paper is organised as follows: Section II provides some background of WSs dynamic reconfiguration in SOA; Section III presents DREWS model; Section IV briefly described DREWS support tool and the tool evaluation; Section V presents evaluation of DREWS and its tool using expert review, and finally, Section VI provides concluding thoughts.

## II. BACKGROUND

According to [5], the system needs to modify its structure in runtime due to changes that occurs either due to expected or unexpected situations. In a service-based system where it is composed of a collection of WSs, the possibility of system changes during runtime is higher because of several reasons, such as due to the unavailability of WSs [6]. As a result, the modification is inevitable and dynamic reconfiguration is required to handle WS replacement. [7] stated that the dynamic reconfiguration is performed due to several reasons such as a WS that cannot accomplish its tasks due to unreachability, business changes, violation of the service level agreement (SLA) or a switch between different WSs' versions.

In regards of different situations, that may request for dynamic reconfiguration, the process may take place at three different levels in a service-based system based on its requirements as described by [8]: business level, service composition level and infrastructure level. In business level with the growing business needs and the expansion of business areas, business processes may need to be reconfigured. In addition, adaptation is also required when some WSs are violating SLA between the service provider and the service requester. In service composition level, the system may need to change dynamically to stipulate new requirements that derived from the business level or new constraints from the WSs and the infrastructure level. Finally,

in infrastructure level, state of resources, such as networks and processors, is considered by WSs execution engines to ensure availability of the resources and energy consumption.

Dynamic reconfiguration at *service composition level* is crucial because it has direct changes to the service-based system structure during runtime [5]. The middleware is a minimum communication abstraction layer that could provide an efficient mechanism to handle flexible composition and heterogeneous WSs and to supports the specification of QoS-based execution properties and temporal characteristics. The dynamic reconfiguration process at service composition level inherited the WS adaptation lifecycle that comprises of three stages: *pre-adaptation*, *in-adaptation* and *post-adaptation* stages. *Pre-adaptation* stage is also known as adaptation preparation stage where environments and the WS's attributes are prepared before adaption process begins, for example selecting a new WS to be used during service reconfiguration [9]. The *in-adaptation* stage is where the adaptation process is actually being performed, for example service reconfiguration. In this stage, there are two prerequisites to fulfil before the adaptation process is possible, first of all on-going processes have been executed or terminated, and second, incoming processes have been stopped [7]. These are to prevent failure operation during the adaptation process.

During the *in-adaptation* stage the systems do not react to any requests, this period is also known as the blackout period. Thus, it is important to handle the blackout period to ensure predictable processing time. The *post-adaptation* is the final stage where the changes during adaptation process are being verified to ensure all the changes can work appropriately. In this stage, necessary actions are taken whenever the adaptation process encounters any errors such as rollback changes due to adaptation failure. Rollback is a mechanism to ensure continuous WS availability by having an ability to return to the previous WS6. In addition, restoration mechanism is crucial to restore data or requests that exists in the previous WS to the WS service after *in-adaptation* stage is completed [6, 10].

The dynamic reconfiguration process is either executed without any external human intervention (also known as self-adaption) [11], or with human intervention (also known as human-in-the-loop adaptation). In self-adaptation, all adaptation steps, decision and actions are performed by the service-based system autonomously. This also assumes that all necessary mechanisms to handle adaptation strategies are built into the system.

### III. THE SOLUTION

DREWS is a middleware-based model to support dynamic reconfiguration of WS. DREWS consists of three main processes: *Manage Adaptation Process (MAP)*, *Selection Process (SP)* and *Reconfiguration Process (RP)*. The three processes are supported by *Connection and Log Recorder (CLR)*, a repository that holds reconfiguration data as shown in Figure 1. The DREWS model underlies dynamic reconfiguration process with the three main tasks: WS selection, WS replacement, and WS verification. WS selection is a task in *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 aspect and QoS are primary concerns to further constrain and select the best WS for a valid WS reconfiguration. WS

replacement is a task in *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 *post-adaptation* stage to verify the proper binding of replacement WS to service-based systems.

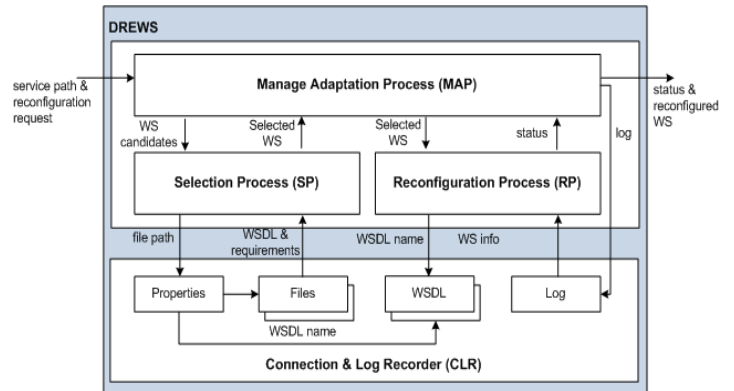


Figure 1: The DREWS model

The following section discusses the DREWS three main processes in details.

#### A. The Manage Adaptation Process (MAP)

MAP is responsible to communicate with a service-based system that requests for dynamic reconfiguration service. The MAP has two roles, first is to regulate tasks between entities such as SP and RP, and second is to verify reconfiguration status after the complete configuration of WS. The MAP is interacting with SP and RP processes while performing tasks of receiving adaptation request, receiving validation feedback, verifying reconfiguration status and releasing web service. Figure 2 illustrates the MAP.

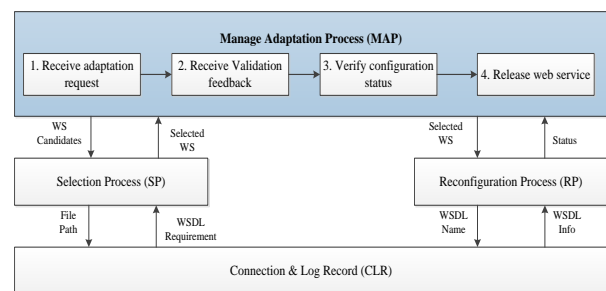


Figure 2: The manage adaptation process (MAP)

The MAP has four steps as follows:

##### 1) Receive adaptation request

A service-based system sends a reconfiguration request to the MAP to conduct dynamic reconfiguration service. Each request contains two inputs: *service path* and *reconfiguration request*. *Service path* is a directory of service that contains information related to WS current connectivity address, WS functional requirements and QoS determined by the service requester and WS candidates that represented by its WSDL URL and log file. *Reconfiguration request* is an initial request that hold value either *Service Failure (SF)* or *Service Upgrade (SU)*. Subsequently, MAP processes the request by submitting WS candidate paths to the SP to perform WS selection.

2) *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 WSDL URL of the selected WS. Then, MAP invokes the execution of RP. This invocation requires WSDL URL of the chosen WS as an input to the RP in order to replace the existing WS with the chosen WS.

3) *Verify reconfiguration status*

After WS has been reconfigured in RP, the MAP receives a reconfiguration status from RP which indicates either success or fail status. Based on the status, the selected WS connectivity is checked to ensure that it is connected properly to the service-based system.

4) *Release web service*

In this step, MAP releases system blocking and send reconfiguration result status (success or fail) to the service-based system. In the event of SU adaptation request status and failed reconfiguration result, DREWS will rollback the connection to the previous WS. However, in the condition of SF adaptation request status and failed reconfiguration result, DREWS will send failure status to notify service requester and reconfiguration connection has to be manually inspected. Finally, in the condition of SF or SU adaptation request and success reconfiguration result, DREWS releases the configured WS to the environment while system is running.

B. *The Selection Process*

SP is a crucial process to find and validate a suitable WS replacement for dynamic reconfiguration. The new WS is selected from a set of WS candidates registered in the CLR. Dynamic service environments cause some difficulties in service selection. Two important factors are considered, i.e. functional requirement and QoS to find the suitable new WS [12]. Functional requirement is a criterion related to WS operation such as a WS to calculate shipment cost and others. Meanwhile, QoS is a criterion to support WS in performing its operations such as the ability of WS to respond during peak hours in less than 0.5 second for calculating shipment cost. The SP is conducted at the pre-adaptation stage of dynamic reconfiguration process. SP consists of four main steps described as follows (refers Figure 3):

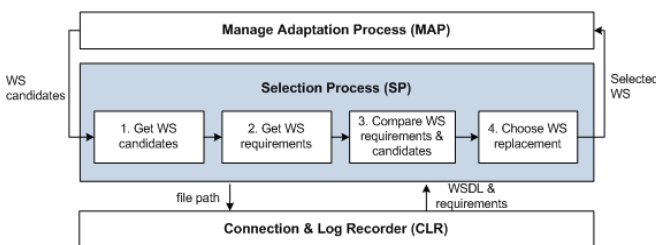


Figure 3: The selection process (SP)

1) *Get WS candidates*

When receiving WS validation requests from MAP, selection process starts to get information of the WS candidates. This information is accessed from the CLR. The WS candidates registered in the CLR areas are based from the subscription agreements between service requester and service provider.

2) *Get WS requirements*

The reconfiguration requirements are retrieved from the

CLR. The requirements that consist of functional and QoS aspects are determined by service requester. This information is maintained dynamically without affecting other reconfiguration information such as WS candidates' information.

3) *Compare WS candidates and WS requirements*

This is a crucial step in SP where WS candidates that represented by WSDL file are compared with WS requirements. There are two types of requirements which are compared: functional requirements and QoS. The WS candidates' functionalities are expressed as WS operations in WSDL files, while reconfiguration functional requirements are established in requirement file by service requester. Each of the operation is compared by using the information retrieved from the files. If one of the required operation is not being provided, the WS candidate will not be used. In addition, DREWS considers four QoS criteria in choosing WS replacement from the WS candidates. The QoS criteria are *service reputation*, *response time*, *availability*, and *throughput*. QoS information is included to the existing WS candidates WSDL. Thus, DREWS has to extend the WSDL file metadata files to include the QoS [13, 14]. The QoS comparison starts after the functional requirements are compared. Each of QoS criterion from extended attribute in the WSDL file is compared with a minimum value from the requirement file. The QoS value should be at least equal to the value in the requirement file. If one of the QoS criteria is not being fulfilled, the WS candidate is not going to be considered as a replacement. The results of the comparison may end with a list of possible WS for replacement. Thus, total score of overall QoS values for each WS candidates is calculated and prioritized in descending order (high to low) to show its achievement.

4) *Choose WS replacement*

The step is to deliver the most suitable WS based on the scoring values of the WS candidates. A WS with the highest score is selected for reconfiguration process. Finally, the new selected WS is sent to the MAP as the SP end result.

C. *The Reconfiguration Process*

The RP main purpose is to conduct service reconfiguration during runtime. The process occurs during in-adaptation stage which requires a support from CLR. RP consists of four main steps described as follows (refers Figure 4):

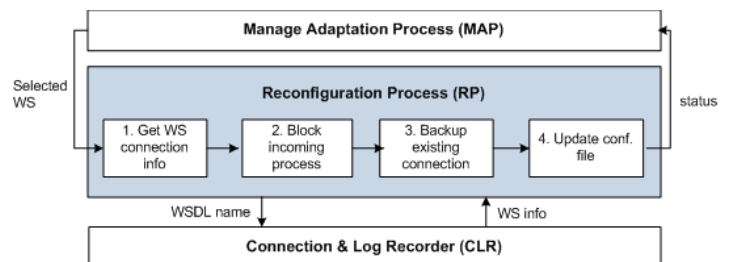


Figure 4: The reconfiguration process (RP)

1) *Get WS connection info*

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. There are two types of information utilised by RP from the WSDL: the WS URL and WS operations.

### 2) 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 reconfiguration process. The precondition to carry out this step is RP has to ensure all ongoing processes inside the existing WS are completed or terminated to prevent operation failures when the existing WS is in blocking mode [10].

### 3) Backup existing connection

In this step, the existing WS connection is copied, backed-up and stored into CLR. The purpose is to handle reconfiguration failure status when upgrading service request is submitted. This means the previous version of WS is still available to be used as a temporary WS.

### 4) 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. After updating the information, RP returns a reconfiguration status of either *success* for success reconfiguration or *fail* to MAP for reconfiguration failure.

### D. The Connection & Log Recorder

The CLR is a repository that used by the DREWS to retrieve and record the WSs information for reconfiguration service purpose. The CLR stores list of WS candidate specifications, WS configuration information, WS requirements, and reconfiguration service history. The CLR is located separately from the DREWS to ensure the service requesters can manage the CLR dynamically without affecting the DREWS main structure. There are five main functions of the CLR.

#### 1) Storing WS path file

This file is a parent file where it is used by DREWS to call all other files stored in CLR.

#### 2) Storing WS configuration information

WS configuration information resides in a *service configuration* file with the aim to minimize connection dependency between service-based system and WSs. The file is updated by DREWS during the reconfiguration process.

#### 3) Storing WS requirements

WS requirements that consist of functional requirement and QoS is recorded in CLR. The information could be updated anytime by the service requester without affecting the WS operations.

#### 4) Storing WS candidates

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

#### 5) Logging reconfiguration activities

The entire process of service reconfiguration is stored in a log file. Both service requester and service provider are able to access the file. This allows both of them to track the overall reconfiguration activities and identify any problems if occurred

### E. The DREWS Attributes

DREWS has to supports two main attributes to perform dynamic reconfiguration of WS. The following sections discuss the DREWS attributes.

#### 1) Functionalities Validation Attributes

WS WSDL XML file contains information about WS parameter, data connectivity, binding, functionalities, and message exchange protocol. It acts as an interface to invoke WS from the service-based system. The main purposes of WS WSDL during the dynamic reconfiguration service is to support SP and RP processes in DREWS. One of the purposes of a WSDL file is it is used to validate WS functionalities. WS functionalities are represented by service operation in WSDL. For example, in Figure 5, *calculatePackage* is the operation to calculate the shipping cost. The validation is conducted by comparing the WS operation name with the functional requirements that have been set by service requester. In this example, *calculate* is the keyword that has been set by service requester in the requirement file.


```
<wsdl:operation name="calculatePackage">  Service operation
  <wsdl:input message="impl:calculatePackageRequest" name="calculatePackageRequest">
  </wsdl:input>
  <wsdl:output message="impl:calculatePackageResponse"
    name="calculatePackageResponse">
  </wsdl:output>
</wsdl:operation>
```

Figure 5: WSDL functionality attribute example

#### 2) QoS Validation 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 the crucial part in WS where it determines user satisfaction when using the WS. Therefore, DREWS has included the QoS attributes by extending WSDL file to include QoS descriptions. There are four main QoS attributes that are frequently considered in the WS selection as highlighted in [13, 14]: *service reputation*, *availability*, *response time*, and *throughput*. WSDL description is extended in this case to helps service provider to provide QoS information of provided WS and service requester could use this information to validate and select WS. DREWS supports the four main QoS attributes described as follows:

- i. Service reputation. Reputation of WS is evaluated by service requesters who previously used the WS. It shows rating of the WS based on user experiences. The higher value indicates good reputation service.
- ii. Availability. The attribute is to ensure the WS is available in their location or the WS is available when required. WS with higher value attribute indicates a better availability of service.
- iii. Response time. Service requester must ensure that the new WS response time is better or at least similar with the existing WS. This attribute indicates a WS has better response time when it has lower response time value.
- iv. Throughput. When selecting a new WS, the system must be able to receive many requests for its operation simultaneously without affecting the WS performance.

This attribute indicates a WS has better throughput when it has higher value.

The QoS attributes described in extended WSDL file are specified by value, offered, unit and direction as follows:

- i. Value. Value of QoS attribute are based on SLA between service provider and service requester. The value is represented using number, e.g. 6
- ii. Offered. QoS attributes availability where the value is either true or false.
- iii. Unit. Measurement unit of QoS e.g. user/millisecond
- iv. Direction. The direction of value, whether increasing or decreasing. Each QoS has its specific direction, e.g. for response time, the lower value (decreasing) indicates WS has better response time.

Figure 6 shows an example of QoS attributes specification of a WS. Finally, the four QoS total achievement is calculated using the Equation (1) (refers Figure 7).

```
<qwsdl:criteria>
<responseTime value="10" Offered="true" unit="msec"
direction="decreasing" />
<throughput value="22" Offered="true" unit="user/sec"
direction="increasing"/>
<availability value="22" Offered="true" unit="%"
direction="increasing" />
<reputation value="22" Offered="true" unit="%"
direction="increasing"/>
</qwsdl:criteria>
```

Figure 6: QoS extension on WSDL example

$$QoS\ Achievement = (RTR/TR) + (T/TR) + (A/AR) + (R/RR) \quad (1)$$

Legends:

- RTR = Response Time Requirement, RRT = Response Time
- T = Throughput, TR = Throughput Requirement
- A = Availability, AR = Availability Requirement
- R = Reputation, RR = Reputation Requirement

Figure 7: QoS achievement equation

#### IV. THE DREWS TOOL SUPPORT

In this section, the tool support underlying the DREWS model is discussed. The tool is developed using JAVA Enterprise Edition API and Apache CXF Open Source Service Framework [15]. The tool consists of four main components which provide dynamic reconfiguration service executor feature and supported by a file repository (refers Figure 8).

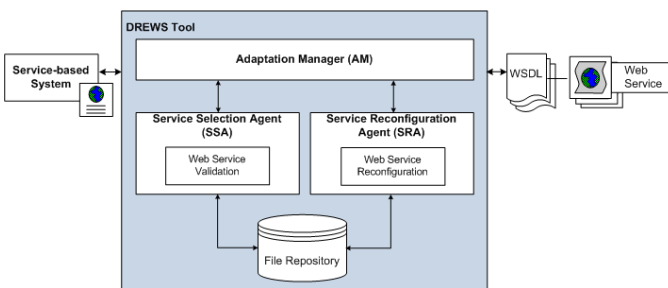


Figure 8: The DREWS tool architecture

The four components are described as follows:

- i. Adaptation Manager (AM). A component to interact with service-based system and WSs. This component distributes, manages and monitors the overall dynamic reconfiguration process.

- ii. Service Selection Agent (SSA). A component to find and validate the new most suitable WS to be adapted.
- iii. Service Reconfiguration Agent (SRA). A component to conduct reconfiguration service during runtime by replacing existing WS with the new selected WS from SSA.
- iv. 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. The data in the file repository support the entire process of the DREWS model.

There are five store types in file repository to support DREWS as shown in Figure 9 (as discussed in Section 3). While Figure 10 shows the screenshot of the tool dynamic reconfiguration logging window.

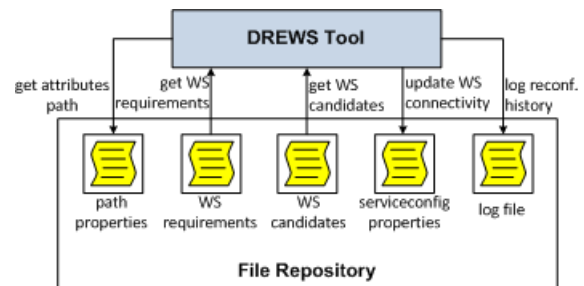


Figure 9: The DREWS file repository

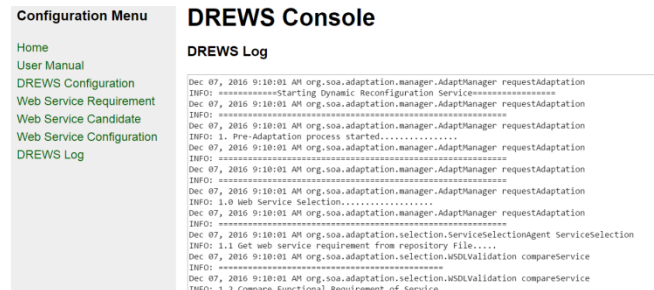


Figure 10: The DREWS logging screen

#### V. EVALUATION

An expert review was an evaluation process involving experts in providing reviews based on their expertise and experience. Expert review is an evaluation approach by allowing analysts to observe a system with more concern to its functionalities, usability and performance [17]. In this work, the evaluation involved five SOA experts from industries to evaluate the effectiveness of the DREWS model and its tool support. The experts have between 5 – 10 years of experience. The expert evaluation process included a number of steps: preparing review protocol, choosing experts, inviting them to take place in the evaluation, evaluating the model by using a scenario and lastly, preparing results of the evaluation based on the feedback from questionnaire given to the experts.

The objectives of the evaluation were:

- i. to check the correctness of finding suitable WSs based on functional and QoS requirements.
- ii. to check the accuracy of error handling in dynamic reconfiguration process.
- iii. to demonstrate the basic features of DREWS.
- iv. to demonstrate the tool is able support automatic

reconfiguration during the runtime stage.

#### A. The Scenario

The Courier Online System (COS) is a service-based system that handles courier shipment daily operations from ordering shipment, checking shipment cost, tracking delivery and various other courier services for their users. The COS was composed by a number of possible independent WSs that was available in the network that performed the desired functionalities of the system. COS system choreographs and coordinates three different services into a work flow to establish the business processes: *locating courier office*, *calculating shipment cost* and *tracking courier* (refers Table 1). In addition, the COS considered relevant QoS aspects in delivering these services to their customers. The COS used Apache CFX to interact with the DREWS tool that independently separates the COS with WS dynamic reconfiguration settings.

Table 1  
COS web service descriptions

WS	Description	QoS Requirements			
		RT.	TP.	A.	Rep.
Office locator service	to helps customers to find an office location in each state around country	10 msc	10 user/sec	80%	10 %
	Pricing delivery service	22 msc	10 user/sec	22%	22 %
Tracking service	to allows customers to monitor their shipping status by inserting their tracking id	70 msc	1000 user/sec	80%	60 %

#### B. Result and Discussion

In the review process the experts were given the COS scenario and being requested to use the DREWS tool to conduct dynamic reconfiguration of the COS's WSs. The scenario is divided into three stages pre-, in- and post-adaptation stages where each stage a set of tasks is assigned to the experts. For each of the tasks, two scenarios were established to address success or failure situation. The results were recorded that indicate whether both of the scenarios were able to be handled by the tool or otherwise. The experts were provided with a logging screen to understand status of the given tasks.

In the pre-adaptation stage, first the experts encountered a successful scenario where they were able to find the best WS that fulfilled requested functionalities and QoS to carry out the reconfiguration. Next, the experts encountered failure to find a new suitable WS. The experts agreed the tool was able to handle the failure by returning a useful error information for their reference.

In the in-adaptation stage, all experts agreed the tool was able to backed-up existing connections and block incoming request before the WS reconfiguration started. During the reconfiguration service, the tool was able to replace the existing WS with the new selected WS that obtained from pre-adaptation stage. After the process was completed, a success status is returned. The experts also acknowledged the tool able to control blackout time that enables dynamic reconfiguration process being terminated when the reconfiguration time was greater than blackout time. During the failure situation, the experts agreed the tool could provide

meaningful error messages for their references and send failure status to adaptation manager (AM).

Finally, in the post-adaptation stage, the experts agreed the tool able to invoke the new WS connection and released it to the real environment. Nevertheless, the expert agreed when encountered with failure to fulfil service upgrade request the tool was able to rollback connection to the prior WS. In addition, the experts agreed the tool provides a sufficient error handling mechanism to handle failure during invocation either to fulfil *service upgrade* or *service failure* request.

In summary, the evaluation was conducted by allowing the professional SOA developers to review DREWS tool. The tool has successfully support dynamic reconfiguration of COS WSs with minimal human intervention during runtime without the need to restart the server. The overall results of the expert reviews are shown in Table 2.

Table 2  
Expert review results

Stage [1]	Feature	Expert				
		1	2	3	4	5
Pre-adaptation	Get WS candidate	√	√	√	√	√
	Get WS requirement	√	√	√	√	√
	Validate FR	√	√	√	√	√
	Validate QoS	√	√	√	√	√
	Get selected WS	√	√	√	√	√
Pre-adaptation	Sufficient error handling	√	√	√	√	√
	Backup existing connection	√	√	√	√	√
	Block Incoming request	√	√	√	√	√
	Replace WS connection	√	√	√	√	√
	Sufficient error handling	√	√	√	√	√
Pre-adaptation	Manage blackout time	√	√	√	√	√
	Validate new WS connection	√	√	√	√	√
	Rollback mechanism	√	√	√	√	√
	Sufficient error handling	√	√	√	√	√

## VI. RELATED WORKS

This section we analyse and compare DREWS with six existing works that support dynamic reconfiguration of WS in SOA. The related works are compared in using seven characteristics discussed in [19] as shown in Table 3.

The comparison showed that existing middleware focused on the partial part of the adaptation process. For example, CoBRA [8] and SASSY [11] focused on the dynamic process during in- and post-adaptation. Other works like LLAMA [20], MLB [22] and iLAND [6] focused in pre- and in-adaptation stages. While, the DREWS focuses in the process occurring during pre-, in- and post-adaptation stages. The selection process conducted during pre-adaptation stage allows the system to select and get the best WS for reconfiguration service-based on its requirements. This selection helps to minimise error during the reconfiguration process due to WS incompatibility.

The second comparison is service selection characteristic indicates most of the middleware focused on functional requirements during service replacement without paying

attention to QoS requirements. LLAMA, MLB and iLAND were three works that attempt to address both functional and QoS requirements. In addition, these works including SASSY provides a tool support to facilitate the replacement of WSs with new most suitable WSs during runtime without human intervention (also known as self-adaptation). As for DREWS, it supports both functional and QoS requirements and requires no human intervention for service replacement.

In the fourth comparison of fault service handling, LLAMA and iLAND were capable of addressing multiple fault services. For example, the feature to handle multiple fault services allows LLAMA to prioritise the WS replacement requests. For example, if there are more than one reconfiguration requests, each of the request priority is calculated and the utmost critical request is attended in sequence. Other works such as CORBA, SOA with OSGi [21], SASSY and MLB focused on single fault service.

In conducting WS reconfiguration, the DREWS adapts

SASSY concept to block incoming requests during the reconfiguration process. It only allows the start of reconfiguration process when all running operations inside the WS have been executed [11]. This feature is adopted in the DREWS to prevent operation failures during the service reconfiguration process. For handling reconfiguration failures, the DREWS adopts CoBRA rollback mechanism. The rollback mechanism allows the DREWS to return to its previous WSs connection when reconfiguration failure is detected during the process [8]. This mechanism is only applicable for service upgrade request. While, for service failure request, the DREWS provides error handling by recording the error messages for user’s reference.

Overall, this comparison proves that DREWS comes out with improvement processes that help service-based system to replace its WSs with a new most suitable WSs during runtime without human intervention.

Table 3  
Comparison of DREWS and existing related works by applying the characteristics of dynamic reconfiguration of WSs in SOA

Reviewed Works/ Features	LLAMA [20]	CoBRA [8]	SOA with OSGi [21]	SASSY [11]	MLB [22]	iLAND [6]	DREWS
Adaptation stages	Pre-adaptation	√	×	×	√	√	√
	In-adaptation	√	√	√	√	√	√
	Post-adaptation	×	√	×	√	×	√
Service selection criteria	Functional requirements	√	×	×	√	√	√
	QoS requirements	√	×	×	√	√	√
Automation	Self-adaptation	√	×	×	√	√	√
	Human-in-the-loop	×	√	√	×	×	×
Fault service	Single service	×	√	√	√	×	√
	Multiple services	√	×	×	×	√	×
Restoration management	×	√	×	√	×	×	√
Rollback mechanism	×	√	×	×	×	×	√
Blackout handling	×	√	×	×	×	√	√

Legend: √ Supported × Not supported

## VII. CONCLUSION

This paper described DREWS, a middleware-based model to improve the abilities of service-based system to replace or reconfigure their WS during runtime. DREWS supports pre-, in-, and post-adaptation stages of dynamic reconfiguration of WS with three main processes and a repository. A tool support underlying the DREWS is designed and developed using Java technologies and Apache service framework. An evaluation is conducted with SOA experts was conducted to evaluate the effectiveness of the DREWS and its tool. The expert review results were promising, as the empirical validation has proven that DREWS supported the dynamic reconfiguration process effectively and automatically. Next, we plan to carry out DREWS second evaluation using an experiment approach to measure its effectiveness compare to another works.

## ACKNOWLEDGMENT

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] W. T. Tsai, M. Malek, C. Yinong, and F. Bastani. “Perspectives on service-oriented computing and service-oriented system engineering,” in *2006 Second IEEE International Symposium on Service-Oriented System Engineering (SOSE’06)*, 2006, pp. 3-10.

[2] J. Fang and Y. Liu. “Research of dynamic SOA collaboration architecture,” in *2009 WASE International Conference on Information Engineering*, 2009, pp. 471-474.

[3] Y. Chen, X. Li, L. Yi, D. Liu, L. Tang, and H. Yang. “A ten-year survey of software architecture,” in *2010 IEEE International Conference on Software Engineering and Service Sciences*, 2010, pp. 729-733.

[4] V. Andrikopoulos, A. Bucchiarone, E. Di Nitto, R. Kazhamiakin, S. Lane, V. Mazza, and I. Richardson. “Service Engineering,” in *Service Research Challenges and Solutions for The Future Internet*, M. P. Papazoglou, K. Pohl, M. Parkin, and A. Metzger, Eds. Berlin, Heidelberg: Springer, 2010, pp. 271-337.

[5] M. G. Valls, I. R. Lopez, and L. F. Villar. “iLAND: An enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems,” *IEEE Trans. on Industrial Informatics*, vol. 9, no. 1, pp. 228-236, 2013.

[6] J. L. Fiadeiro and A. Lopes. “A model for dynamic reconfiguration in service-oriented architectures,” *Software System Model Software & Systems Modeling*, vol. 12, no. 2, pp. 349-367, 2012.

[7] F. Irmert, T. Fischer, and K. Meyer-Wegener. “Runtime adaptation in a service-oriented component model,” in *Proceedings of the 2008 International Workshop on Software Engineering for Adaptive and Self-managing Systems - SEAMS '08*, 2008, pp. 97-104.

[8] F. André, E. Daubert, and G. Gauvrit. “Distribution and self-adaptation of a framework for dynamic adaptation of services,” in *The Sixth International Conference on Internet and Web Applications and Services (ICIW)*, St. Maarten, Netherlands Antilles, 2011, pp. 16-21.

[9] S. Shrivastava and A. Sharma. “An approach for qos based fault reconfiguration in service oriented architecture,” in *2013 International Conference On Information Systems and Computer Networks (ISCON)*, 2013, pp. 180-184..

[10] H. Goma, & K. Hashimoto. “Dynamic self-adaptation for distributed service-oriented transactions,” in *2012 ICSE Workshop On Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2012, pp. 11-20.

[11] M. P. Romay, L. Fernández-Sanz, and D. Rodríguez. “A systematic review of self-adaptation in service-oriented architectures,” in *The*

- Sixth International Conference on Software Engineering Advances, Barcelona, Spain, 2011, pp. 1-7.
- [12] Y. Gong, L. Huang, F. Jiang, and K. Han. "An approach to web service dynamic replacement," *International Journal of Grid and Distributed Computing*, vol. 7, no. 1, pp.1-12, 2014.
- [13] V. Agarwal and P. Jalote. "From specification to adaptation: An integrated qos-driven approach for dynamic adaptation of web service compositions," in *2010 IEEE International Conference On Web Services (ICWS)*, 2010, pp. 275-282.
- [14] H. Gao and H. Miao. "A quantitative model-based selection of web service reconfiguration," in *2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2013, pp. 365-371.
- [15] Apache CXF, Apache CXF: An open-source services framework, Available at <http://cxf.apache.org/>, 2016.
- [16] D. Sjoeborg, J. Hannay, O. Hansen, V. Kampenes, A. Karahasanovic, N. Liborg, and A. Rekdal. "A survey of controlled experiments in software engineering," *IEEE Transactions on Software Engineering* *IEEE Trans. Software Eng.*, vol. 31, no. 9, pp. 733-753, 2005.
- [17] B. Soufi. "Survey and expert evaluation for e-banking," in *International Conference on Human Interface and the Management of Information*, 2013, pp. 375-382.
- [18] R. Ilahi, N. Admodisastro, N. Mohd. Ali, and A. B. Sultan. "Dynamic reconfiguration of web service in service-oriented architecture," in *Proc. of the Int. Conf. on Computational Science and Engineering (ICCSE)*, Center of Excellence in Semantic Agents (COESA), 2016, pp. 165-170.
- [19] K. Lin, M. Panahi, and Y. Zhang. "The design of an intelligent accountability architecture," in *Proceedings of the IEEE Int. Conf. on E-Business Engi.(ICEBE'07)*, 2007, pp. 157-164.
- [20] H. Lv, W.Liu, and H. Zhang. "The application and research of a dynamic architecture for service based on SOA," in *Proceedings of the 2009 International Conference on Information Engineering and Computer Science*, 2009, pp. 1-4.
- [21] V. Krishnamurthy and C. Babu. "Dynamically reconfiguring services in soa applications: a pattern-based approach," in *EuroPLOP '12 Proceedings of the 17th European Conference on Pattern Languages of Programs*, Irsee, Germany, 2012, pp. 1-13.