Enhancement of Organizations Performance Based on Modified Web Services Model

Adeeb Mohammed Hider Al-qershi

Faculty of Administrative Sciences, Taiz,University Yemen. Email: adeeb@taiz.edu.ye

Abstract— Nowadays, Information Systems (IS) are the corner stone in modern organizations to perform their IT and Business processes. Most of these systems not mutual, so Web Services (WS) are very important technique used to satisfy this mutuality. Actually, WS provides an ability of increasing the performance of organizations services. This enables organizations to enhance their services by publish them in standard vision due to exchange them in universal form. The WS quality depend on many metrics such as Response Time(RT) and CPU Utilities. Furthermore, delivery data from original provider into customer should be reliable and confident. In this paper Local Client ache(LCC) and Delivery Checker (DC) are proposed as additional efficient components into traditional WS model. These components have been implemented and tested to enhance WS quality. Based on experiment's results on these components, they proved efficiency based on reduce RT and CPU Utilities of WS. LCC increased the user interactivity with WS applications as well as working with WS without internet connection (offline mode). On the other hand, the DC has been ensured delivery notifications to consumer without loss by apply many of efficient precautions. Modified WS Model utilities are much promised and show the superiority of the proposed model. `

Index Terms— IS, IT, Web Services, WS, Response Time, RT, DC, CPU Utilities

1- Introduction

Organization's services must be available in consumer hand efficiently. Acquiring WS from original server takes a more time based on Internet speed, Internet traffic, networks troubleshooting and server availability. Thus, there is needing to set WS available each time that user request. This mechanism increases the performance of IT and business services. Subsequence, consumers save contact with the organizations. In addition, WS enhance IT-business services availability and accessibility among several departments in the organizations. Applying WS reduce the cost and response time of services and makes them available for invocation time [1]. Web Services (WS) traditional model has three components, which are WS Provider, WS Consumer and WS Registry as in [4]-[2].Thus, the accessing to WS should be done from WS original resource/Server.

2- Literature Review

A web service: " is a software system designed to support interoperable machine-to-machine interaction over a network"[2].WS is a model that allow organization to publish their services online [7]. There are two models for implement WS, which are SOAP and REST [3].

2.1 WS Traditional Model

As shown in Fig. 1, the WS standard model has three components: service provider, service registry and consumer.

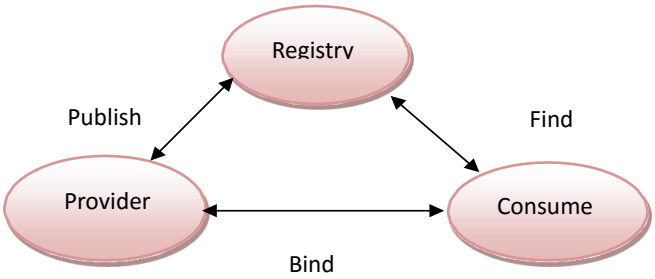

Fig. 1: WS model entities [4]

Practically, service provider creates his own service(s), then the services(s) must be published on registry hosting, which must be has additional information about provider such as address and port type. After that the service will be available for consumer [4] Web cashing Techniques

There are many techniques that used web cashing for enhancing web utilities. A generic WWW caching system

[6] as in Fig. 2 ,based on The first hit of client will be on proxy cache, but if hit fail it will be redirect to main Web server [6].

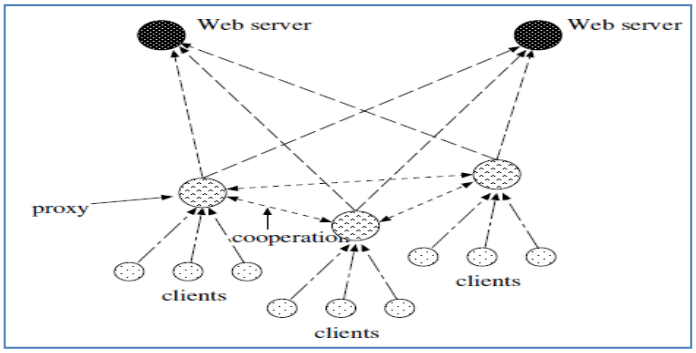

Fig. 2: A generic WWW caching system [6]

There are many caching strategies refer to [8]. These strategies represented in:

Lazy Loading: depends on loading data to cache only when necessary.Write Through: adds or updates data to cache whenever database updated

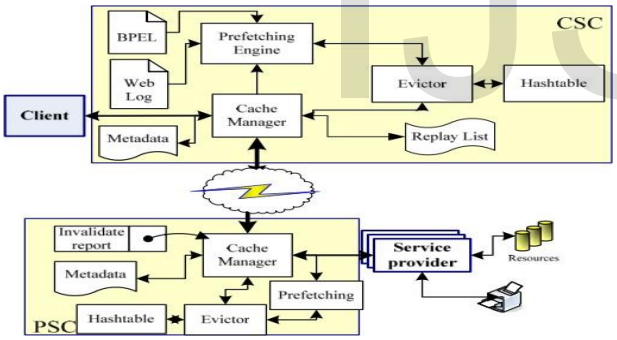

Fig. 3: Dual cache architecture [10].

Adding TTL (Time To Live): here the time sets in seconds until the key is expired. Dynamic Object Cache Service model to cache object as dynamic key and value called CacheDOCS [39] Dual cache architecture [10] for WS as in Fig. 3 bellow. This model fits with PDAs devices only [10] and argues that using dual cache save data from loss, but I think it's not enough to ensure delivered data to client was without loss.

3- THEORETICAL FRAMEWORK

In this section, a detail description of new proposed model of Web Services is demonstrated. Actually, this includes some extensions and modifications to the traditional Web Services Model. The suggested model has been developed by supplement of many effective components. The purpose of the suggested components is to improve the performance of WS. Consequentially, the proposed model allows more efficient management for request and response of Web Services within organizations, as well as for Web Services providers.

To assess the proposed model performance, vital measurements factors have been used. These factors such as: Response Time and CPU Utilization. These factors will clarify the degree of performance

enhancement of the targeted organizations.

3.1 PROPOSED WS NEW COMPONENTS

The new enhancement WS proposed model as in Fig. 4 has additional effective components that have been added into client side WS as listed below:

1. Client Cache Manager (CCM)
2. Delivery Checker (DC)
3. Local Client Cache (LCC)

The Figure bellow demonstrates the proposed modified WS model.

3.2 PROPOSED WS NEW COMPONENTS

The new enhancement WS proposed model as in Fig. 4 has additional effective components that have been added into client side WS as listed below:

1. Client Cache Manager (CCM)
2. Delivery Checker (DC)
3. Local Client Cache (LCC)

The Fig. bellow demonstrates the proposed

Modified WS model

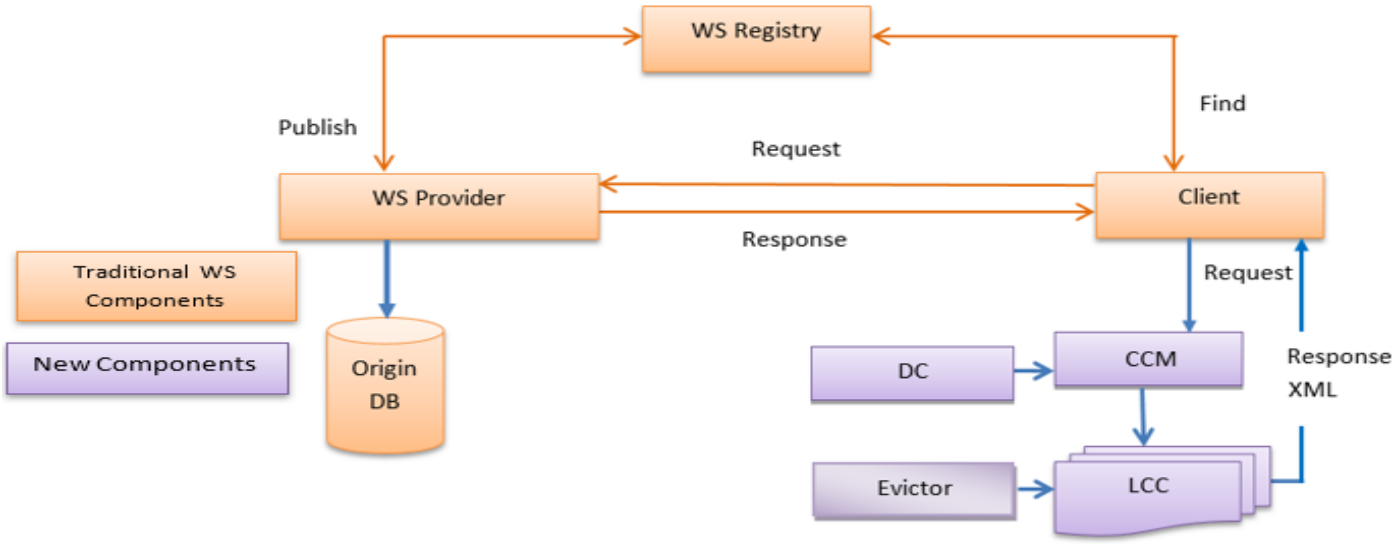

Fig. 4: The proposed modified WS modified Model

3.3 Client Cache Manager (CCM)

3.4 CCM Functionality

CCM has been designed to achieve many of processes, which are demonstrated bellow.

3.4.1.1 Create Local Client cache LCC

3.4.1.2 Receive updated data from server as XML file.

3.4.1.3 Update Local Client Cache(*LCC*) by assign delivered XML file from WS Provider by do these steps

3.4.1.4 Move data from delivered XML file into LCC

3.4.1.5 After step (1) done successfully then delivered XML file will be deleted to save memory space. Also, the modified WS model has only one LCC.

3.4.1.6 Achieve searching operation on LCC and return data.

3.4.1.7 Redirect the search process to WS server, when the local cache search hit miss.

3.5 CCM Contribution

I. Reduce dependency on the server by perform all process in client side, thereby network bottlenecks and server load will be reduced automatically

II. Access Web Services offline as possible.

III. There is only one updated (LCC), which saves memory space

IV. Reduce response time

3.6 CCM Algorithms

Algorithm: *Update Local Cache*
Input Data : *notifyFilePath , LCCFilePath*
Result : *Updated LCC file*

```
1  begin
2     deliveredFile=loadXmlFile(notifyFilePath)
3     LCC= loadXmlFile(LCCFilePath)
4     nodes = deliveredFile .getXMLNode(operationType)
   /* search for operation node which is insert, update or delete*/
5  for each node in nodes  do
6     if node.value= 1 then /* 1 = insert operation */
7        insertIntoLCC()
8     else if node.value= 2 then /* 2 = update operation */
9        updateLCC()
10    else
11       deleteFromLCC()
12    end for
13       saveXMLFile(LCC)
14 end
```

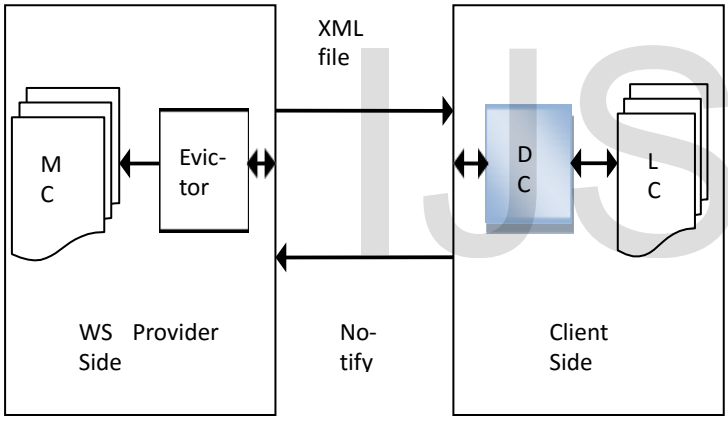

Fig. 5: DC functionality in modified WS Model

This algorithm shows how Local Client Cache(LCC) is updated from delivered notification file .**First**, Loading both LCC and delivered notifyFile into **delivered File** and **LCC** objects .**Second**, get xml node that holds type of notification operation, which is insert ,update or delete operation.**Third**, loop during all notifying file, if node value equal ***1*** then **insertIntoLCC** method will invoked to insert record(s), or if the node value will be **2** then **updateLCC** method will invoked to update *LCC*, else the operation will be delete by invoking **deleteFromLCC** method. **Finally**, *saveXMLFile* method saves all previous modifications into *LCC*.

4 DELIVERY CHECKER (DC)

Delivery Checker: its Modified WS Model component within Client Side that has been developed to check if notifications have been delivered successfully from WS Provider to client.

4.1 DC FUNCTIONALITY

This component ensures that the consumer gets all interested notifications without loss by achieving these steps

i. Get notify from server for ready notifications.

ii. Receiving delivered notifications.

iii. Check if the XML notifications file doesn't lose any data by comparing count of received records with count of records in Server Meta Cache.

iv. Check if notifications file has been reflected correctly on the local cache.

v. Check if 3, 4 steps done successfully, then send notify to evictor in Server side to evict those notifications, otherwise, evictor doesn't evict the data from MC and data within MC will be available any time.

Fig. 5 shows DC functionality among modified WS model components

4.2 DC CONTRIBUTION

There are many scenarios, which show possibilities of losing delivered XML file:

i. Loss delivered data due to internet or network interruption. 2- Loss delivered data due to unknown storage destination.

ii. Loss delivered data due to limited disk space.

4.3 DC ALGORITHM

Algorithm: *Delivery Checker*
Input Data: userid, datafile, count, wsName, deliverState
Result: update local cache, if datafile of notifications has sent complete and local cache updated successfully then Meta Cache will be evicted

```
1. begin
2.   datafile= invoke → WS(getData(userid))
3.   if exist (datafile) =true then
4.     if empty(datafile)=false then
5.       count=getRecordsCount(datafile)
6.       wsRecordCount= invoke→WS(countMetaCacheRecords(count,wsName,userid)
7.         if count= wsRecordCount then
8.             if updateLocalCache()= count then
9.                 deliverState ← 1
10.            else
11.                deliverState← 0
12.            end if
13.        else
14.            deliverState← 0
15.      end if
16.    else
17.        deliverState← 0
18.  end if
19.  else
20.      deliverState← 0
21.  end if
22.  if deliverState=1 then
23.      invoke→ WS(evictMetaCache(wsName , userid, deliverState,currentDate)
24.      message("Received Successfully")
25.  else
26.      invoke→ WS(evictMetaCache(wsName , userid, deliverState,currentDate)
27.      message("Failed Receive")
28.  end if
29. end
```

5 EVICTOR

Evictor: it's a modified WS model component has been used to evict the leased used data from LCC based on modified LFU algorithm. LFU more suitable with web objects rather than other evicts algorithms as LRU [13].

5.1 EVICTOR FUNCTIONALITY

The evictor is applied the general idea of LFU (Leased Frequency Used) algorithm, which based on evict the least access objects [11], which has been developed to replace least frequency data with new one.The proposed solution to indicate Leased used records as well as fits with XML files is adding new tag to each record and this tag represents the Last Activity Date. Consequently, the records with less active will be deleted. The strategy of replacement in modified WS model applies LFU, when the LCC size reaches to *N* size.

5.2 EVICTOR CONTRIBUTION

Evictor contribute within modified WS model as:

1. Save LCC space
2. Reduce RT

6 EXPERIMENTS AND RESULTS

In this section many experiments have done to evaluate impact of new WS model components. This impacting have been done to test RT and CPU utilization factors.

6.1 Measure Response Time (RT)

Response Time: is the vital factor in any system/application computing [12]. RT According to the IBM Dictionary of Computing is "The elapsed time between the end of an inquiry or demand on a computer system and the beginning of a response" [5]. RT is used to measure time from submit request until get first response [13].

Experiment 1: *Measure LCC vs. WS RT of batch search*

This experiment has been conducted for 100 times, each time is performed on 1000 record. In this experiment, evaluation of the performance depending on RT is done. A comparison shows the difference between RT of batch search for random selected records in LCC and WS provider database.

Input: 100 search operations as Random Batch search on XML file with 1000 record.

Result: **comparison** between RT of LCC and Web Service Batch search

Time Unit: **Seconds**

Experiment Details: This experiment has been performed for 100 search operations from 1100 record. This search is done by batch 100 search operations, where the records are selected randomly between 1 and 1100 student records. The result is shown in

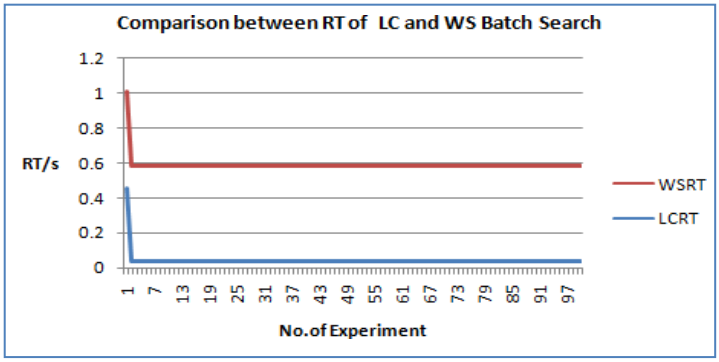

Fig. 6.

Fig. 6: Measuring RT for Batch search from LC and WS

Result Analysis:

The previous Figure clarifies the performance of LCC with RT. There is a big time gap between search from LC and search from WS server using WS. It has been noticed that the LCC enhanced the WS performance significantly.

Experiment 2*: Measure LCC vs. WS RT of Manually search*

This experiment has been conducted 100 times from 1000 records. These records have been selected and achieved randomly on LCC and WS from WS Provider, bellow Fig. illustrates the results.

Input: 100 search operations as Random manually search, XML file.

Result: comparison between RT of LCC and Web service using manually search

Time Unit: Seconds

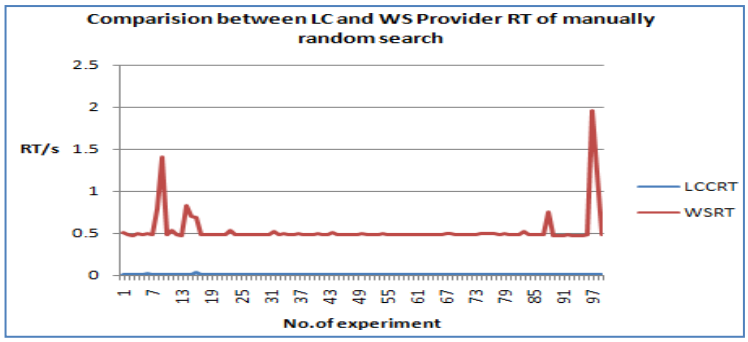

Fig. 7: Measuring RT for Manualy search from LC and WS

Result Analysis:

From the previous Fig., the reducing of RT with LCC is noted compared ith RT with WS. This proves that LCC is enhanced the WS performance.

6.2 Measure CPU Utilization Factor

CPU Utilization indicates extend of CPU Utilization to perform the processes [14].

Thus, CPU Utilization refers to CPU Usage [15].Practically; efficiency of the proposed model implies high CPU Utilization.The experiments are done here to measure extend impacting of WS on CPU Utilization factor.

Experiment 1: *LCC vs.WS CPU Utilization of Manually search*

In this experiment, CPU Utilization has been conducted for 100 times, each time is performed in about 1000 record. This test has been done on search from LCC and WS. The results appear in Fig. 8**.**

Input: 100 search operations as Random manually search, XML file

Result: comparison between CPU Utilization of Local cache and Web Service Batch search

Time Unit: Seconds

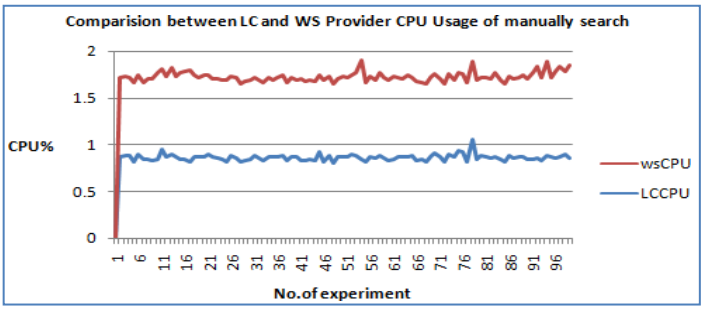

Fig. 8: Results of CPU Utilization of LCC and WS manual randomly search Result Analysis

7 RESULT ANALYSIS

Based on previous Figure the CPU Utilization of searching operations has been reduced, when they are done on LCC compared to WS CPU Utilization.Based on previous Fig. the CPU Utilization of searching operations has been reduced, when they are done on LCC compared to WS CPU Utilization.

Experiment 2: *LCC vs.WS CPU Utilization of Batch search*

In this experiment, CPU Utilization have been tested in batch search .This experiment has been done 100 times, in each time is performed on about 1000 record. This test done to search from LCC and WS the results appear in Fig. 9.

Input: 100 search operation as Random batch search, XML File.

Result: comparison between CPU Utilization of Local cache and Web service Batch search.

Time Unit: Seconds.

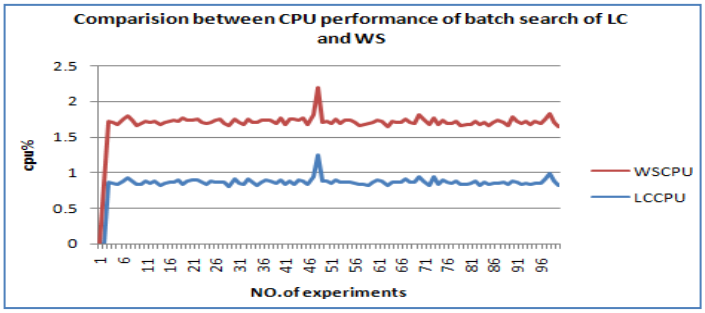

Fig. 9: Measurement result of CPU Utilization of LCC and WS of randomly batch search.

Refer to Figure previous of CPU Utilization with LCC in batch search is reduced compared with WS batch search. From previous Figures of RT measurements, the bellow Figs.10, and 11 illustrate both RT and CPU Usage in both manually and batch random search of client operations.

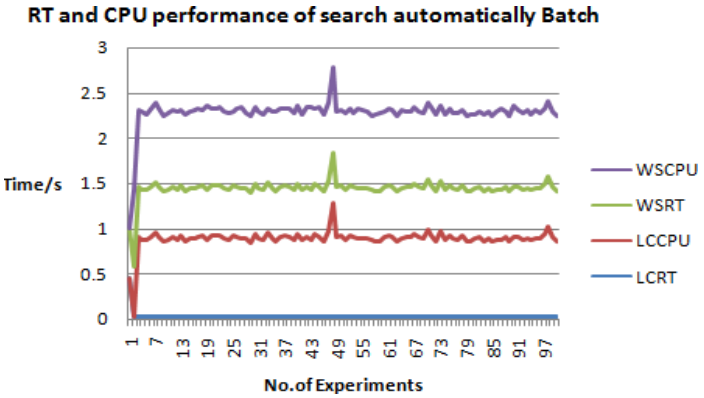

Fig. 10: Show RT and CPU Utilization in manually search from Both LCC and WS

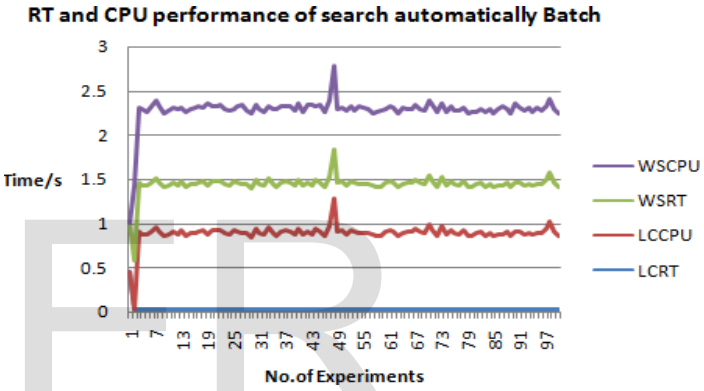

Fig. 11: Show RT and CPU Utilization in Batch search from Both LCC and WS

8 CONCLUSION

In this paper, the proposed ***WS*** has been developed and tested successfully. With proposed ***WS*** the organizations that have interesting in WS can meet their demands based on ***this new model***. The new ***WS model*** has many efficient components include: *CCM, LCC* and *Evictor*. ***The proposed WS model was*** underwent several experiments to evaluate its performance. Based on results analysis, it is noted ***The proposed WS*** saves the significant system resources by reducing RT, CPU Utilization. For instance, LCC decreases working with WS server and makes WS available offline as possible. Moreover, the ***interactivity*** between consumers/users and WS client side applications was increased.

REFERENCES

[1] N.Sasikaladevi and Dr.L.Arockiam, "*Optimizing the Performance of Dynamic Composite Web Service Using Caching.*", IEEE, 2011.

[2] Connolly, T., C. Begg, and R. Holowczak, "*BUSINESS DATABASE SYSTEMS*.", Pearson Education Limited ,2008.

[3] *Abilio*, C.M.G.a.R., "*Systems Integration Using Web Services, REST and SOAP: A Practical Report.*FSMA", 2017. **19**: p. 34-41.

[4] Dustdar, S. and W. Schreiner, "*A survey on web services composition",* Int. J. Web and Grid Services, 2005. **1**(1).

[5] IBM. "*Response Time*. ",2017 [cited 2018; Available from: http://searchnetworking.techtarget.com/definition/response-time

[6] Bhuvaneswari, G.R.K., "*Semantic web service discovery for mobile* web *services.",* Int. J. Business Intelligence and Data Mining, 2018. **13**.

[8] *Tsalgatidou*, A. and Pilioura, T. "*An Overview of Standards and Related Technology in Web Services.",* Kluwer Academic, 2002. **12**.

[9] Leymann, F., "*Web Services: Distributed Applications without Limits".* IBM Software Group.

[10] Abeywickrama, D.B., et al., "*Web Services Foundations*. ",2014: Springer.

[11] Alexis Huf, I.S., Frank Siqueira, "*Planning and execution of heterogeneous service compositions*.", 2017.

[12] Craige V.MCMeruray , A.T.W., Vadim Meleshuk, Marke E. Gabarra, "*Request-Specefic Authentication for WS Resources*. ",2017: USA.

[13] Microsoft," *Asynchronous Programming With Async and Await. "*,2018.

[14] Waleed Ali , S.M.S., and Abdul Samad Ismail," *A Survey of Web Caching and Prefetching.",* ICSRS, 2011. **Vol. 3, No. 1**.

[15] Inc, T. "*Response Time*.", 2017 [cited 2017; Available from:https://www.techopedia.com/definition/9181/response-time.

[16] Microsoft. "*Interpreting CPU Utilization for Performance Analysis*.", 2018 [cited 2018; Available from: https://blogs.technet.microsoft.com/winserverperformance/2009/08/06/interpreting-cpu-utilization-for-performance-analysis/.