I.J. Information Engineering and Electronic Business, 2015, 4, 31-38
Published Online July 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijieeb.2015.04.05

Assessing the Behaviour of Web Services using Finite States

Maheswari S
SCSE, VIT University, Chennai 600127 TN, India
Email: maheswari.s@vit.ac.in

Justus Selwyn
SCSE, VIT University, Chennai 600127 TN, India
Email: justus.s@vit.ac.in

***Abstract*—**Web service are the technology of a choice when developing business applications that needs to be loosely coupled, platform independent and capable to cross enterprise boundaries. The interactions that occur between web services need to be captured because such interactions would be very useful if captured using appropriate structures and analyzed for various purposes such as assessing the responsiveness of a web service to complete peer's requests. Since the invocations of web services (WS) are dynamic, the behaviour of the WS will be dynamic depending on how the invocations discover, and get serviced by WSs. For this reason if the states of the behaviour of a WS can be captured and assessed, then tuning the WS and its performance improvement can be engineered at any stage. This work is presented in this paper.

***Index Terms*—**Web Services, Runtime Behaviour, Finite States, SOA, Behaviour Capture.

I. Introduction

Web service provides a standard means of communication between the different software applications giving the ability to access different services in a unified and in interoperable manner over the internet. Each service can offer various options for quality characteristics based on the technical specifications like performance, availability, reliability, scalability and so on.

They are basically termed as contracts in the context of web service applications which verifies the run time behaviour of the systems [2]. Any Web application requires interaction between the client and the server on multiple occasions to retrieve and submit data.

This requires page submission or navigation. Using Web Services, we can accomplish two things: avoid page submits and provide the ability to consume Web services [5]. Avoiding page refreshes enhances the browsing experience for the end user by making pages load faster.

The web service behaviour enables client-side script to invoke remote methods exposed by Web Services, or other Web servers, that support the SOAP and Web Services Description Language (WSDL). This behaviour provides developers the opportunity to use and leverage SOAP, without requiring expert knowledge of its implementation [4]. The web service behaviour supports the use of a wide variety of data types, including intrinsic SOAP data types, arrays, objects, and XML data [2]. The web service (WS) behaviour is implemented with an HTML Component (HTC) file as an attached behaviour, which can be used in Microsoft Internet Explorer 5 as well as in later versions.

Service Oriented Architecture(SOA) : Web services can be thought of as components that can be described , published, located and invoked over the internet and all these activities take place in run time. The functional architecture of web services is given in Fig. 1. In order for web services to be able to work together these principles are integrated to be service oriented architecture (SOA) [7]. The SOA organizes the web services into three basic roles:

- *Service Provider*
- *Service Requester*
- *Service Registry*

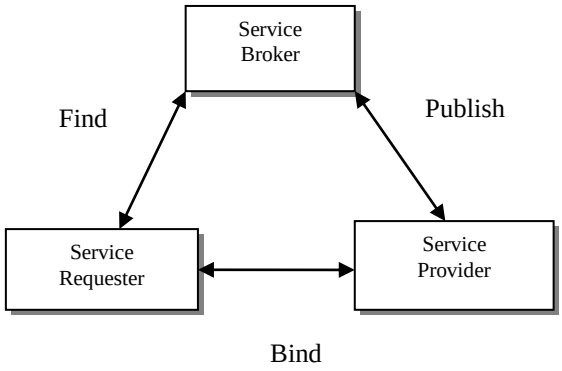


Fig. 1. Web Services Architecture

The four functional components of SOA are

- *Service Implementation*
- *Publication*
- *Discovery*
- *Invocation*

Service Implementation: There are two basic approaches to building a web service: building from scratch or provide a wrapper to an existing application or service so that it exposes a web service interface.

Publication: The web service description document written in WSDL where the document describes what the web service will do where it can be found and how to invoke it.

Discovery: Once the web service appears in registry any application can discover the service and therefore locate it.

Invocation: There are two steps in invoking the web service i.e. by using a simple object access protocol (SOAP) where the WSDL file downloaded contains the information needed to create a client using SOAP or the client creates a SOAP message describing what it wants the remote web service to do.

A. Motivation

With the advent of technology internet is becoming an essential component in every day's life. The day to day activities which a common man needs like railway reservations, airline reservations, e-commerce, e-governance, e-learning and so many other services are just a click away. However, in the pool of web services, the invoker has not gained much clue to choose which, when, and why web-service to get better serviced.

To answer these questions, the invoking application or the end-user has to have a better knowledge on the behaviour of the web-service, in terms of its performance, functionality, its states and the final quality of service output [7], [8]. In order to provide the invoker with better knowledge about the web-service, we have to provide it's behaviour patterns. Thus we landed up in this work, to capture the runtime behaviour of web-services.

This is possible because, in various scenarios a web-service may behave differently according to its nature of invocation, and will exhibit external behaviour states under each scenario of invocation [5]. These external behaviours are captured using the Finite State Machine principles, and are named as different states, based on its status quo, waiting, execution, completion states [18]. Hence this work on capturing the runtime behaviour of web-services.

B. Outline of the Work

Section II discusses the behaviour analysis of the web service (WS) in relevance to the states. In section III the architecture for capturing the behaviour of the WS is proposed and some results of preliminary experiments conducted. Section IV presents main experimental work and the results, following the conclusion and future works in section V.

II. Behaviour Analysis

The captured behaviour of web servicers (WS) are used in many ways of analysis to build better strategies in WS design, tuning, and improved functionalities and performances [6]. The analysis we have opted is to build a knowledgebase (KB) of WS's behaviour patterns with respect to time, infrastructure and resource utilization. The KB is then subject to interpret to take decisions or a feedback providing repository for opting to choose a WS.

There are four different kinds of behaviour in web services

- Service behaviours enable the customization of the entire service's run-time including the service host base [9].
- Endpoint behaviours enable the customization of service endpoints and their associated endpoint dispatcher [11].
- Contract behaviours enables the customization of both the client run time and dispatch run time in the client and service applications [14].
- An operation behaviour enables the customization of the client operation and dispatch operation classes again on the client and the service [6].
- Behaviour compatibility in web services can lead to the termination of a web services composition if they are not detected before the actual execution [11].

These categories of behaviours address the aspects of WS from the WS developers' perception, from the service providers' perception and from the invoking application/service's perception. In this work we have opted to

- Propose set of States and its Transitions for Web Services and the checks that each state undertakes
- Conduct preliminary experiments to understand the timer state, which will help assess the behaviour of WS with respect to time
- Conduct a case-study, and evaluate the dynamic behavioural states of each invocation

The dynamic behaviour of Web services regarding the development of new services and constantly changing existing ones require a continuous evaluation process, leading to capture web service information with respect to their quality and performance evaluation [18], [19].

III. Capturing the Behaviour using FSM

In this work we have tried to identify the various aspects of a web service and categorizes these requirements into various states, which will be finite [1]. Based on the basic SOA, we have opted to propose different states of a web-service (WS) and its transition events and check. Using these states a state transition diagram can be prepared which helps us to find the various normal and the alternate flows while running a web application [9]. Thus the knowledge inferred from the above run time behaviour of the model can be treated as an intermediary before executing further applications.

A. Proposed System Architecture

The overall objective of this work is to capture the behaviour of a WS and take that to a knowledgebase, which can infer dynamic decisions on the credibility and performance related issues of a WS. Fig. 2., shows the proposed system.

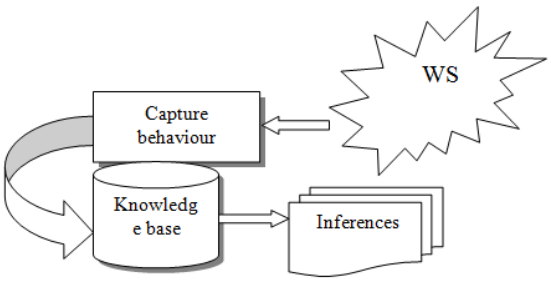


Fig. 2. Capturing the WS behaviour for a Knowledgebase

Module 1: In this module a sample web service has been developed using dotNet programming language. The demo application created is an e-commerce web site which allows a customer to login while entering the web site.

We have also tried to invoke different web-services from different service providers. Invoking them and analyzing their behaviour based on the timer state is an attempt to assess the run-time behaviour of WS. The preliminary experiments on WS with Timer State are presented in sub-section B.

Module 2: In this module, the run-time behaviour of the web-service is captured as collections of "state transitions". Each transitions are represented as states using FSM principles, which helped us to derive the necessary inferences.

Module 3: Using the Case Study we have taken in Module 1, the runtime behaviour is analyzed and concluded on inferences – which can help us determine the nature, performance and quality of the web-service under consideration. The case-study we have taken is an online banking portal.

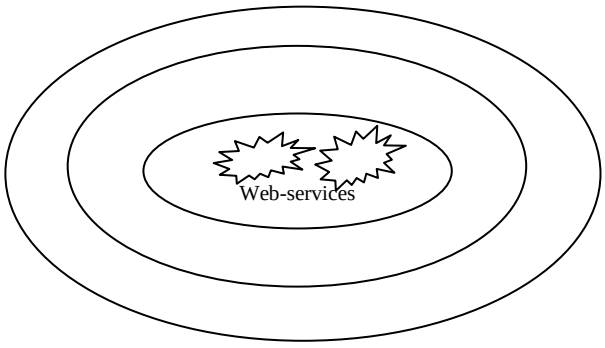


Fig.3. Web-Services and SOA Layer

B. State Transition Diagram

Capturing web service behaviour: Web services are basically loose coupled system which is used for dynamic binding of services. Therefore for efficient run time execution it is necessary to identify both the normal and alternate flows. Hence we have used state transition diagrams which represent the various states for generalised web applications.

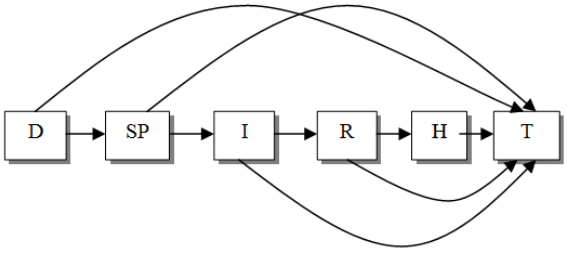


Fig.4. Behaviour State Transition Diagram

D: Discovered State:

The service provider publishes the web service (WS) in the UDDI registry. The service requester requests the service broker for the specified web service. The service broker responds to the client request with the details of the web service and the URL of the WS using WSDL file, if available in the UDDI. The requestor then binds to the service provider with the model key and the URL given by the broker.

Once the WS is bounded to the requester, and goes to the 'D' state. This is the first state, where the WS has made its presence in the UDDI registry in the web.

SP: Service Provider Monitoring Agent

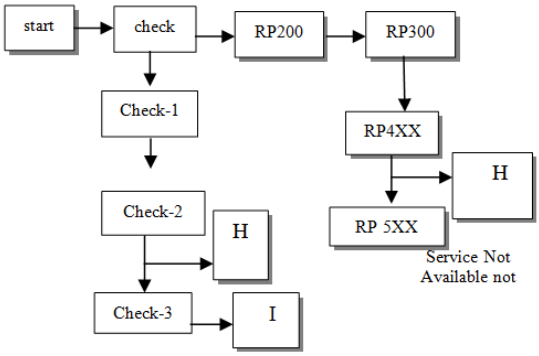


Fig. 5. Service Provider Monitoring

The server machine responds to the client machine with a response code. The state maintains the code of the server response and identifies the action accordingly. The agent checks with the hardware requirements of the client. The agent verifies the network speed of the system. After successful completion of check-1 through check-3, the WS goes to the next state, Invoked.

I: Invoked State

Once the Check 3 is passed, the state of the web-service is changed to Invoked state. The invoke state has the following check states as show in Table 1. Check to see whether a specific, relevant method is invoked, and proceed for input parameters checking, availability of cache memory allocated for this invoked service/method are sequentially made. If check 4 fails, then the WS is pushed to the previous state SP. The flow of sub-states in

'I' invoked state is sequential when these checks return to be true.

If check-7 fails, WS goes to the 'H' state, and if check-8 fails, WS goes to 'SP' state, otherwise, after check-9, WS goes to 'R' the running state.

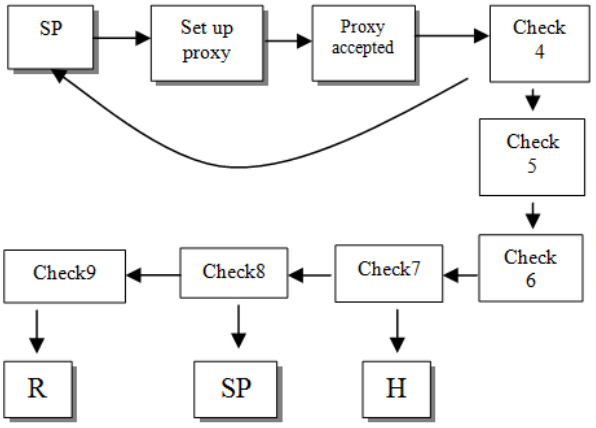


Fig. 6. Invoked state cycle

R: Running State

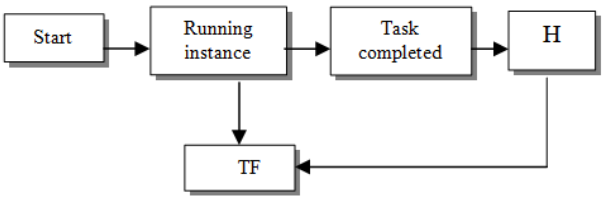


Fig.7. Running State of a WS

Table 1. Web-service State Checks

Check 1	Check for the hardware requirements
Check 2	Check for the network speed
Check 3	Check for the client system error
Check 4	Check for the method invoked
Check 5	Check for the input parameters
Check 6	Check the technologies used
Check 7	Internal cache memory
Check 8	Check for the conversation id if any
Check 9	Check for the idempotent links if any

A web-service is said to be in a running state when has become a running instance after passing out all the nine checks that are given in table 1.Once the task gets finished it goes to the halt state (H). The WS goes to the TF state if it encounters any runtime technical failures. This running state uses the resources allotted to it for execution. While in execution if a web-service runs of resources, memory out-run, network not-found, or invalid input parameters- then the web-service goes to TF state, and returns the control to the calling function or page

TF: Technical Fault State

A running instance of WS may get into the technical faults (TF) state because of the following reasons:

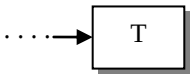
- Client Crashes
- Timer Expiry
- Session Expired
- Server Fails
- Input/Output Stream exceptions
- Kill Command

T: Timer State

Each of the state is connected to a timer to find out the time spent on the state and the transition time of a particular state to another state. The time spent by a WS in each state is calculated and assessed for any bottlenecks that might lead to any performance issues. The transition time between states indicates the intermediate requirements of the WS to complete its task.

The total time taken = time spent on the state + the transition time between the states.

Time spent on the state: this includes execution time + invoking other services/resources



Timer state is kept at the end because it gives the results of the time taken by the WS to complete its functions right from invocation to termination.

C. Preliminary Experiments

Some preliminary experiments were conducted to validate the Timer State (T). The results authenticate that the Time (t) spent by the WS will have a say on its behaviour on the time complexity. However, calculating time complexity is out-of-scope of this work, and we tried to check the 'time taken' as a whole factor.

We considered to invoke four difference Web-services (WS), and invoked them in 10 iterations, with different sets of input. We could see differences in the time taken for each iteration. The WSs are from two different service provider "webservicex.net" and "servicerepository.com". The details of the WSs are:

WS1: Currency Converter
WS2: Calculator
WS3: Temperature Converter
WS4: Stock Quote

Table 2. Currency Converter: Webservicex.net

Iteration	Description	Server Response	Time taken in millisecs
1	USD-INR	61.6499	14469
2	EUR-USD	1.1602	9496
3	INR-UAH	0.2569	1300
4	WST-USD	0.4064	968
5	BZD-BRL	1.3035	877
6	KPW-USD	0.0011	914
7	NAD-LBP	130.9939	880
8	LBP-EUR	0.0006	1216
9	ZAR-INR	5.3342	6145
10	TWD-ZAR	0.3677	10803

Average Time taken for this webservice: 4706 ms

Table 3. Calculator: ServiceRepository.com

Iteration	Web service input (Two numbers)	Webservice Output	Time taken in millisecs
1	74,43	117	809
2	39,63	102	927
3	32,59	91	974
4	26,64	90	743
5	54,47	101	630
6	28,98	126	620
7	43,52	95	533
8	21,80	101	849
9	68,59	127	686
10	64,25	89	844

Average Time taken for this webservice:761 ms

These services are invoked from two different service providers Webservicex.com and ServiceRepository.com

The experiments objective is to calculate the time spent by the service to serve the request. We invoked each of the services separately in 10 iterations with different inputs each time. Each iteration recorded different time duration in milliseconds spent by the web-service in finishing the tasks. The results are shown in Tables 2, Table 3, Table 4, and Table 5.

Table 4. Stock Quote: Webservicex.net

Iteration	Quote Input	Webservice Output	Time taken in millisecs
1	A	A-Last: 38.16;1/21/20154:02pm;Change: +0.23;Open: 37.75;High: 38.41;Low: 37.68;Volume: 2721609; Mrk: 12.796B; Prev.Close: 37.93;Perc Change: +0.61%; Ann Range: 35.6223 - 43.2833; Earns: 1.4925.46; Company: Agilent Technolog	939
2	N	N-Last: 105.01;1/21/20154:02pm; Change: -0.55;Open: 104.96;High: 106.75;Low: 104.21; Volume: 394269;Mrk: 8.058B;Prev.Close: 105.56;Perc Change: -0.52%;Ann Range: 69.48 - 120.77;Earns: -1.254;P-E: N/A;Company: Netsuite Inc Comm	1006
3	Y	Y -Last: 445.47;1/21/20154:01pm;Change: -1.61;Open: 447.78;High: 451.97;Low: 442.19; Volume: 54071;Mrk: 7.174B;Prev.Close: 447.08;Perc Change: -0.36%;Ann Range: 361.01 - 482.00;Earns: 45.0019.93;Company: Alleghany Corpora	983
4	O	O -Last: 52.75;1/21/20154:02pm ;Change: +0.01;Open: 52.60;High: 52.96;Low: 52.27; Volume: 2117414;Mrk: 11.747B;Prev.Close: 52.74;Perc Change: +0.02%;Ann Range: 39.25 - 53.12;Earns: 0.98353.65;Company: Realty Income Cor	989
5	P	P -Last: 15.68;1/21/20154:00pm;Change: +0.01;Open: 15.64;High: 15.95;Low: 15.47; Volume: 4078252;Mrk: 3.263B;Prev.Close: 15.67;Perc Change: +0.06%;Ann Range: 15.26 - 40.44;Earns: -0.111;P-E: N/A;Company: Pandora Media	938
6	Q	Q-Last: 59.78;1/21/20154:05pm;Change: -0.17;Open: 59.86;High: 60.64;Low: 59.57;Volume: 755245;Mrk: 17.640B;Prev.Close: 59.95;Perc Change: -0.28%;Ann Range: 45.25 - 60.79;Earns: 2.574;P-E: 23.29;Company: Quintiles Transna	1001
7	X	X-Last: 22.06;1/21/20154:00pm;Change: +0.48;Open: 21.54;High: 22.42;Low: 21.42;Volume: 7711010;Mrk: 3.209B;Prev.Close: 21.58;Perc Change: +2.22%;Ann Range: 21.39 - 46.55;Earns: 0.669;P-E: 32.26;Company: United States Ste	909
8	G	G-Last: 20.50;1/21/20154:00pm;Change: -0.07;Open: 20.55;High: 20.56;Low: 20.33;Volume: 565391;Mrk: 4.440B;Prev.Close: 20.57;Perc Change: -0.34%;Ann Range: 13.68 - 20.61; Earns: 0.85;P-E: 24.20;Company: Genpact Limited C	903
9	C	C-Last: 47.74;1/21/20154:00pm;Change: +0.48;Open: 47.35;High: 48.23;Low: 47.15;Volume: 22873308;Mrk: 144.6B;Prev.Close: 47.26;Perc Change: +1.02%;Ann Range: 45.18 - 56.95;Earns: 2.911;P-E: 6.24;Company: Citigroup	1204
10	B	B-Last: 35.88;1/21/20154:02pm;Change: +0.48;Open: 35.44;High: 36.04;Low: 35.05;Volume: 205579;Mrk: 1.954B;Prev.Close: 35.40;Perc Change: +1.36%;Ann Range: 29.47 – 30.01;Earns: 2.041;P-E: 17.34;Company: Barnes Group	977

Average Time taken for this webservice: 984 ms

Table 5. Temperature Conversion: ServiceRepository.com

Iteration	Webservice Input in Celcius	Webservice Output in Fahrenheit	Time taken in millisecs
1	50	122	1405
2	63	145.4	940
3	95	203	922
4	20	68	956
5	60	140	959
6	58	136.4	918
7	41	105.8	970
8	82	179.6	1356
9	43	109.4	957
10	39	102.2	910

Average Time taken for this webservice:1029 ms

These results show that the total time taken by a WS includes the identified states D-SP-I-R-H states shown in Fig. 4. Taken a WS the time values remains consistent with

IV. Case Study and Experimental Results

Services in the bank: A bank named “Altius Bank” has created an online banking system for providing better services to the customers without having to be physically present in the bank. Altius Bank is a full-fledged financial service provider founded by first generation professional entrepreneurs. Altius strives to excel in investment banking, debt synchronization which uses Soap based services and encompasses WS-Reliable messaging and

WS-security.

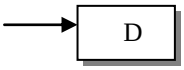
This bank has implemented all its services using SOAP based web services for NEFT and RTGS interbank fund transfer.

User Registration: The user registration has been implemented in such a way that username and ID are associated with a unique PAN card number. A successful registration would create a unique account for the customer.

A. Application of the States in the Case Study

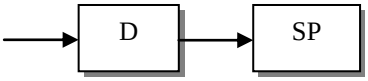
Step 1.

Discovered State:



Input: Request to the UDDI server to get details for Altius Bank.

Step 2



SP- Service provider monitoring check

Table 6. List of HTTP response codes

400	Bad file request
401	Unauthorised
403	Forbidden Access/Denied
404	File Not Found
408	Request timed out
500	Internal error
501	Net implemented
502	Service Temporarily Overloaded
503	Service unavailable

Input : a mouse click on the link if the discovered state is successful.

Operations : It will check for the following response codes if the server didn't respond

Output: Choose the list of services you want to avail.

Table 7. Checks for Invoked State

Check 1	Internet access enabled with 128 bit browser. OS -Win Xp,Vista,7,Mac OS Processor-Pentium or equivalent processor
Check 2	The network speed should be 56.6kbps
Check 3	Client system error
Check 4	Check the method invoked e.g. createAccountRequest() method is used while creating an account
Check 5	Check for the input parameters i.e to enter the user details while creating an application
Check 6	Not needed in all applications
Check 7	▪ In order to use internet banking services we need to enable the java script. ▪ To access some parts of the website it is necessary to install an adobe flash player. ▪ For printing PDF documents the system needs to have adobe reader installed. ▪ Session cookies are security features one of the information used to ensure that some part of the information is confidential. ▪ SSL(secure socket layer) must be enabled.
check 8	Checking the internal cache configurations and a minimum 512MB RAM is required.

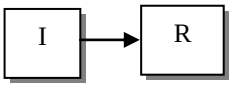
Step 3

This invoked state tries to establish a proxy connection with the bank server. Once the request is accepted the invoking agent checks whether the client system is compatible with the application requirements.

Output: Once the metrics are checked the service is ready to be invoked. In this case the user tries to invoke the create account service and the interbank fund transfer.

Step 4

R: Running State



Input: Call createAccountMethod()
Enter User Details.
Create Account

Step 5

TF: Technical Fault state

Input : logout=true

For several other reasons a running instance can come to halting state. Some of them are:

- For security reasons access to internet banking has been interrupted.
- Request could not be completed because of server problems.
- Site can be slow sometimes if too many users are trying to access the the site at the same time.
- If the client card number is entered incorrectly.
- Make sure that the client is logged in properly.
- Session timed out.
- Successful finished the running instance.

B. Inter Bank Funds Transfer

The funds that is transferred between the banks is through ACH-NEFT/RTGS. The method that implements the interbank transfer service is transferFundsInterBank().When the operation is invoked by the user the bank checks for the available balance and invokes the initiate Transfer() web services. Now when the bank invokes the execute Transfer() web service. This happens for NEFT transfers which will enable the target bank to know that the transfer of funds process is initiated. As the parameters are passed on to this method contains the details of the transferee the target bank after doing due diligence will return the status of the funds transfer.

If there are issues at this point of the transaction the system records this as "failed transaction". Subsequently when the update_status() is invoked with the status as "success" the balance is updated in the respective account. Requests Req_1, Req_2 upto Req_n, are trying to invoke the fund transfer WS of Aliuis Bank. Fig. 9 shows the cluster of requests waiting, and possible behaviour paths the WS would take. We created test requests to the WS to transfer fund to our own account and two other accounts of our colleagues of different banks, and found out each request service have different behaviour patterns.

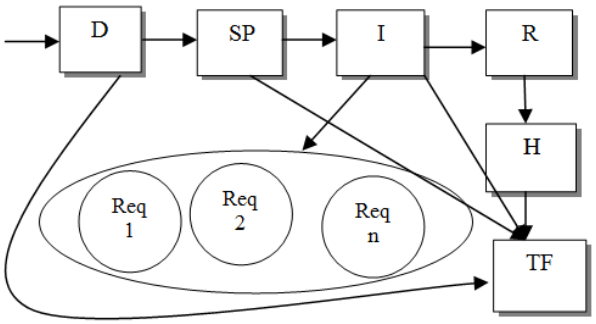


Fig. 9. Requestors for Fund Transfer Services

Req_1 := D – SP – I – R – H
Req_2 := D – SP – T F
Req_3 := D – SP – I – Req_3
Req_4 := D – SP – I – R – H – TF
Req_5:= D – SP – I – R – TF
Req_6 := D – TF
Req_7:= D – SP – I – TF
Req_8:= D – SP – I – R – Req_8

Each of the transaction at different states is tied to a timer state through which we were able to estimate the time spent (in secs) by the WS on that state.

Table 8. Time Spent on Each State

Req	D	SP	I	R	H	TF	Total
1	0.034	0.011	0.025	0.326	0.004	-	0.4
2	0.012	1.000	-	-	-	0.008	1.02
3	0.085	0.762	0.054	-	-	-	0.901
4	0.031	0.043	0.013	0.413	0.006	1.000	1.506
5	0.172	0.078	0.041	0.082	-	1.000	1.373
6	1.000	-	-	-	-	0.007	1.007
7	0.052	0.037	1.000	-	-	0.009	1.098
8	0.047	0.025	0.083	0.316	-	-	0.471

A simple experiment conducted on the time spent by each request on the WS in each of the states has given some insights into the run-time behaviour of the WS of Altius Bank's Fund Transfer. We had the set the timer expiry to 1.000s

Observations: Whenever there was an error through TF or due to some other reasons, the timer expires and the WS moves on to the next state omitting the transaction details. In Req2, from SP to TF state, total time spent is 1.02, but with failed fund transfer. Req1 and Req4 has successful fund transfer with total amount spent 0.400 and 1.506 secs respectively. Other WS have failed due to TF or have returned back to the caller application due to some recursive events.

This initial work on capturing the runtime behaviour of the WS using Finite States has helped us to assess the time spent by WS is different for different requests or calls. Similarly other runtime attributes can also be estimated and a knowledgebase could be created on this, so as to decide which type of requests will be preferred by the calling applications.

V. CONCLUSION

Web services are everywhere, and their dynamisms are tremendous. Every call to WS results in various landing pages with success messages or failure/error messages. Which WS is preferred for my call? This can be answered by knowing the behaviour of web-services in detail. In this work we have proposed an architecture on capturing the run-time behaviour of a WS, and this answers which WS to prefer based on the organization's requirement [17]. A finite set of states are identified which depicts the behaviour of the WS during runtime. The state transition part is also validated using the Fund Transfer WS of Altius Bank. We conducted small experiments to find out the time spent by the WS on each of the states. Though the results are simple, it is convincing that this work will direct us to carry out more empirical works in this area.

ACKNOWLEDGMENT

The author wish to thank the SMEs in Entapp Solutions for giving us space to conduct the experiments and thank the management of VIT University who helped to carry out this base research work

REFERENCES

[1] R. Alur, "Timed Automata", In Proceedings of the 11th International Conference on Computer Aided Verification(CAV'99), Vol. 1633 of LNCS, pages 8–22. Springer-Verlag ", 1999.

[2] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti, "Run-time Monitoring of Instances and Classes of Web Service Compositions", In ICWS '06: Proceedings of the IEEE International Conference on Web Services, pp. 63-71, 2006.

[3] Dainela Remenska, Tim A.C Willemse,Kees Verstoep,Jeff Templon, "Using Model Checking To Analyze The System Behaviour of the LHC Grid", International IEEE/ACM Symposia on Cluster, Cloud and Grid Computing, pp.335-343, 2012.

[4] Hamidi Yahyaoui, Zakaria Maamar, Erbin Lim, Phillipe Thiran, "Towards A Community Based Social Network Driven Framework For Web Services", *Future Generation Computer Systems*, Vol. 29, No. 6, Pages 1363–1377, August 2013.

[5] Jocelyn Simmonds Et. Al., "Runtime Monitoring of Web Service Conversations", *IEEE Transactions on Services Computing*, Vol. 2, No. 3, July-Sept, 2009

[6] Jocelyn Simmonds, Shoham Ben-David, Marsha Chechik. "Monitoring and Recovery of Web Services", *The Smart Internet*, LNCS 6400, pp. 250-288, 2010.

[7] Jocelyn Simmonds, “Dynamic Analysis of Web Services”, Doctoral Thesis, University of Toroto, 2011.
[8] M. Kanmani, I. Suhaimi, K. Mojtaba, M. Keyvan, and T. Sayed Gholam Hassan, "An Evaluation of Process Mediation Approaches in Web Services", Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services, Paris, France, ACM pp. 48-55, 2010.
[9] Alessio El At., “Runtime Monitoring of contract regulated web services”, *Fundamenta Informaticae XXI*, pp. 1001-1017, 2009
[10] Xi Chen, Zibin Zheng, Qi Yu, and Michael R. Lyu, “Web Service Recommendation via Exploiting Location and QoS Information”, *Ieee Transactions on Parallel and Distributed Systems*, Vol. 25, No. 7, July 2014
[11] Guobing Zou, Qiang Lu, Yixin Chen, Ruoyun Huang, You Xu, and Yang Xiang, “QoS-Aware Dynamic Composition of Web Services Using Numerical Temporal Planning”, *IEEE Transactions on Services Computing*, Vol. 7, No. 1, January-March 2014.
[12] Zibin Zheng, Yilei Zhang, and Michael R. Lyu, “Investigating QoS of Real-World Web Services”, *IEEE Transactions on Services Computing*, Vol. 7, No. 1, January-March 2014.
[13] Udhav S. Lahane,K.N.Shedge, “A Review of Web Service Recommendation Systems”, *International Journal of Science and Research (IJSR)*, Volume 3, Issue 11, November 2014.
[14] Zibin Zheng, Hao Ma, Michael R. Lyu, and Irwin King, “Collaborative Web Service QoS Prediction via Neighborhood Integrated Matrix Factorization”, *IEEE Transactions on Services Computing*, Vol. 6, No. 3, July-September 2013.
[15] L. Barakat, S. Miles, and M. Luck, ‘‘Efficient Correlation-Aware Service Selection”, in Proc. IEEE 19th International Conference on Web Services, pp. 1-8, 2012,
[16] G. Kang, J. Liu, M. Tang, X. Liu, B. Cao, and Y. Xu, “AWSR:Active Web Service Recommendation Based on Usage History”,in Proc. IEEE 19th International Conference on Web Services, pp. 186-193, 2012.
[17] Saida Boukhedouma, Mourad Oussalah, Zaia Alimazighi, Dalila Tamzalit, "Service Based Cooperation Patterns to Support Flexible Inter-Organizational Workflows", *International Journal Information Technology and Computer Science*, Vol. 6, No. 4, pp. 1-18, 2014.
[18] Tarek S. Sobh, Medhat Fakhry, "Evaluating Web Services Functionality and Performance", *International Journal Information Technology and Computer Science*, Vol. 6, No. 5, pp. 18-27, 2014.
[19] Abhishek Kumar, Manindra Singh, "An Empirical Study on Testing of SOA based Services", *International Journal Information Technology and Computer Science*, Vol. 7, No. 1, pp.54-66, 2015.

Authors’ Profiles

Maheswari S is a fulltime Assistant Professor in the School of Computing Sciences and Engineering at VIT University. She is pursuing her PhD in the areas of Web-services, Semantic Web, and Knowledge-based Web-services.

With a post-graduate in Computer Technology, she is specialized in Web-Technologies and Service Oriented Architecture. She is a member of ACM-India professional association.

Justus Selwyn is a Doctorate in Computer Science specialized in Object-Relational Modeling and Knowledge Engineering. His areas of research include Software Engineering, Knowledge Engineering, Object-Relational Modeling and Knowledge based System Design.

He has been into academic & industrial research and has published several of his research work results in International Journals and presented some of them in International Conferences – including SwSTE in Israel and DASMA in Germany.

He is a member of IEEE, ISTE, IAENG professional associations. Presently he is working as Associate Professor at VIT University, Chennai, and chairs the Software Engineering Research Group of the University. He has collaborative consultancy works on system design and development with software companies.

How to cite this paper: Maheswari S, Justus Selwyn,"Assessing the Behaviour of Web Services using Finite States", IJIEEB, vol.7, no.4, pp.31-38, 2015. DOI: 10.5815/ijieeb.2015.04.05