Measuring the dependability of Web Services for use in e-Science experiments

P. Li, Y. Chen, A. Romanovsky.

TECHNICAL REPORT SERIES

No. CS-TR-938 December, 2005

TECHNICAL REPORT SERIES

No. CS-TR-938 **December, 2005**

Measuring the dependability of Web Services for use in e-Science experiments

P. Li, Y. Chen and A. Romanovsky

Abstract

This paper introduces a dependability assessment tool (WSsDAT) for Web Services monitoring and testing. It allows users to evaluate dependability of Web Services from the point of view of their clients by collecting metadata representing a number of dependability metrics. This Java-based tool can be deployed in diverse geographical locations to monitor and test the behaviour of a selected set of Web Services within preset time intervals. The graphical user interface of the tool provides real-time statistical data describing dependability of the Web Services under monitoring. The tool makes it possible for the users to identify typical patters of dependability-specific behaviour depending on the time of the day, days of the week or the client locations, or collect and analyse service dependability measurements during long periods of time, as well to obtain dependability-related information about current service behaviour. The paper reports a successful series of experiments on using this tool for assessing two BLAST Web Services widely used in the bioinformatics domain. The paper shows how the tool can be employed by e-scientists to help them in making informed choices improving the overall dependability of the 'in silico' experiments they are performing using a set of distributed services.

Bibliographical details

LI, P., CHEN, Y., ROMANOVSKY, A..

Measuring the dependability of Web Services for use in e-Science experiments
[By] P. Li, Y. Chen, A. Romanovsky.

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Science, 2005.

(University of Newcastle upon Tyne, Computing Science, Technical Report Series, No. CS-TR-938)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE
Computing Science. Technical Report Series. CS-TR-938

Abstract

This paper introduces a dependability assessment tool (WSsDAT) for Web Services monitoring and testing. It allows users to evaluate dependability of Web Services from the point of view of their clients by collecting metadata representing a number of dependability metrics. This Java-based tool can be deployed in diverse geographical locations to monitor and test the behaviour of a selected set of Web Services within preset time intervals. The graphical user interface of the tool provides real-time statistical data describing dependability of the Web Services under monitoring. The tool makes it possible for the users to identify typical patters of dependability-specific behaviour depending on the time of the day, days of the week or the client locations, or collect and analyse service dependability measurements during long periods of time, as well to obtain dependability-related information about current service behaviour. The paper reports a successful series of experiments on using this tool for assessing two BLAST Web Services widely used in the bioinformatics domain. The paper shows how the tool can be employed by e-scientists to help them in making informed choices improving the overall dependability of the 'in silico' experiments they are performing using a set of distributed services.

About the author

Yuhui Chen is a PhD student at University of Newcastle upon Tyne.

Alexander (Sascha) Romanovsky is a Professor in the CSR. He received a M.Sc. degree in Applied Mathematics from Moscow State University and a PhD degree in Computer Science from St. Petersburg State Technical University. He was with this University from 1984 until 1996, doing research and teaching. In 1991 he worked as a visiting researcher at ABB Ltd Computer Architecture Lab Research Center, Switzerland. In 1993 he was a visiting fellow at Istituto di Elaborazione della Informazione, CNR, Pisa, Italy. In 1993-94 he was a post-doctoral fellow with the Department of Computing Science, the University of Newcastle upon Tyne. In 1992-1998 he was involved in the Predictably Dependable Computing Systems (PDCS) ESPRIT Basic Research Action and the Design for Validation (DeVa) ESPRIT Basic Project. In 1998-2000 he worked on the Diversity in Safety Critical Software (DISCS) EPSRC/UK Project. Prof Romanovsky was a co-author of the Diversity with Off-The-Shelf Components (DOTS) EPSRC/UK Project and was involved in this project in 2001-2004. In 2000-2003 he was in the executive board of Dependable Systems of Systems (DSoS) IST Project. Now he is coordinating Rigorous Open Development Environment for Complex Systems (RODIN) IST Project (2004-2007).

Suggested keywords

DEPENDABILITY,
AVAILABILITY,
MEASUREMENT,
INTERNET,
EXPERIMENTS

Measuring the dependability of Web Services for use in e-Science experiments

Peter Li[1], Yuhui Chen[2] and Alexander Romanovsky[2]

[1]Manchester Centre for Integrative Systems Biology,

School of Chemistry, University of Manchester, M60 1QD, UK.

[2]School of Computing Science, University of Newcastle upon Tyne, NE1 7RU, UK.

Abstract

This paper introduces a dependability assessment tool (WSsDAT) for Web Services monitoring and testing. It allows users to evaluate dependability of Web Services from the point of view of their clients by collecting metadata representing a number of dependability metrics. This Java-based tool can be deployed in diverse geographical locations to monitor and test the behaviour of a selected set of Web Services within preset time intervals. The graphical user interface of the tool provides real-time statistical data describing dependability of the Web Services under monitoring. The tool makes it possible for the users to identify typical patters of dependability-specific behaviour depending on the time of the day, days of the week or the client locations, or collect and analyse service dependability measurements during long periods of time, as well to obtain dependability-related information about current service behaviour. The paper reports a successful series of experiments on using this tool for assessing two BLAST Web Services widely used in the bioinformatics domain. The paper shows how the tool can be employed by e-scientists to help them in making informed choices improving the overall dependability of the 'in silico' experiments they are performing using a set of distributed services.

1. Introduction

Research in dependability has traditionally been focused on closed systems where all components or services belong to the same management domain and are composed statically. This allows the integrators to collect prior information about the availability and reliability of those components and to employ architectural solutions that can provide the required level of dependability (Tai *et al.,* 1993; Randell *et al.,* 1995). This approach is not applicable to the service-oriented architectures where services belong to different organisations and the integrators do not have enough information about their dependability characteristics (Ferguson *et al.,* 2003). They need special support to make their decisions using valid and up-to-date information collected on-line. Dependability is, therefore, a major issue in service-oriented environments which are used in e-Science where virtual organisations (VO) are formed on an *ad hoc* basis. From a certain point of view, the organisations whose services are invoked within a scientific workflow can be considered to form a VO during its enactment (Oinn *et al*., 2004). However, the reliability of services can be erratic since various types of service faults may be experienced by users and this can lead to failure during the enactment of the workflow. Faults can occur due to an inability to reach the service because of a network problem. The service might also be inoperative because its server is undergoing maintenance or may have become overloaded with requests. Perhaps more critically, the data output generated by a service may be wrong due to incorrect or corrupted requests and this will have serious consequences to the results of scientific workflows.

Issues with the reliability of services are particularly prevalent in the area of bioinformatics. Academic and non-commercial organisations deploy Web Services for public use by scientists in the life sciences community without any prior service level agreements. Such services are used by scientists knowing of their unreliability despite the fact that they may not always be available for the reasons outlined above (Stevens *et al*., 2004). These services are orchestrated into workflows which represent '*in silico*' experiments, analogous to those performed by experimental scientists in laboratories (Oinn *et al*., 2004). Such *in silico* experiments may be long lived due to the large volumes of data being analysed, whilst there may also be requirements on the timeliness of the workflow enactment. It is therefore essential that such e-Science workflows are executed using the most reliable of services. The possibility of failure can be reduced by selecting those services which are the most reliable based on metadata about the behavioural characteristics of Web Services. Furthermore, the availability of quantitative metadata can provide a means of predicting the reliability of a given service.

In this paper, we describe a tool which was developed to help e-Scientists in improving the dependability of *in silico* experiments that they run over a number of distributed services. We start with a detailed description of the tool architecture and functionality (Section 2). In Section 3 we compare our approach with the existing work in the area of experimental evaluation of Web Services. In the following part of the paper we demonstrate the applicability and usefulness of the tool by discussing the results of our experimental work in analysing the dependability of two BLAST services widely used in the bioinformatics domain (Section 4). In Section 5, we overview typical patterns of how this tool can be used to help the e-scientists and overview our plans for the future work. Section 6 concludes the paper.

2. The WSsDAT tool

Our work on the tool started with formulating the essential requirements which a general Web Services dependability-monitoring tool needs to meet. The main requirement is that such a tool should be able to monitor a Web Service continuously for a preconfigured period of time and record various types of information in order for the dependability of a service to be measured. Firstly, the tool should provide an interface to accept user's inputs and map these user inputs into internal processing actions. Secondly, the tool needs to be able to invoke the Web Service effectively and wait for results; internal and external exceptions should be monitored during this period. When the output of the service invocation is received, the response time for the service should be recorded and analyzed. Ideally the output of the service needs to be assessed to determine whether the Web Service functioned properly and whether it passed or failed according to the scientists' demands. Moreover when the test invocation failed then any fault messages generated by the service should also be documented. If available, these messages will provide insights behind the problems causing the service failure. Finally, the tool should be able to produce reports of the test and monitoring procedures.

2.1 Overview

The requirements of a general Web Services dependability-monitoring tool were realised by the development of a Java-based application called Web Services Dependability Assessment Tool (WSsDAT) which is aimed at evaluating the dependability of Web Services. The tool supports various methods of dependability testing by acting as a client invoking the Web Services under investigation. The tool enables users to monitor Web Services by collecting the following reliability characteristics:

- ***Availability and Functionality:*** Calls are made to a Web Service at defined intervals to check if the Web Service is functioning. The tool is able to test the semantics of the response which are generated by the Web Service being monitored. It is possible to pre-configure the tool using a regular expression which represents the correct response expected by the scientist from a given Web Service and ensure the service is functioning according to that expected by its user. Results returned from a Web Service are recorded for further analysis which can be manually carried out by a user.
- ***Performance:*** The WSsDAT measures the round-trip response time of calls made to the Web Services. Average response time of successful calls is used as performance metric of a Web Service.
- ***Faults and exceptions:*** The tool records any faults generated by a failed invocation of a Web Service. Internal and external exceptions, for example networking timeout exception, are also recorded for further analysis.

Further to the above metadata recorded by WSsDAT, the tool can also be used to test and monitor the dependability of Web Services at geographically disparate locations through the deployment of the tool on different computers. It is important to understand the behaviour of a

Web Service from the point of view of the clients, in order to comprehend the networking consequences between the clients and the Web Service.

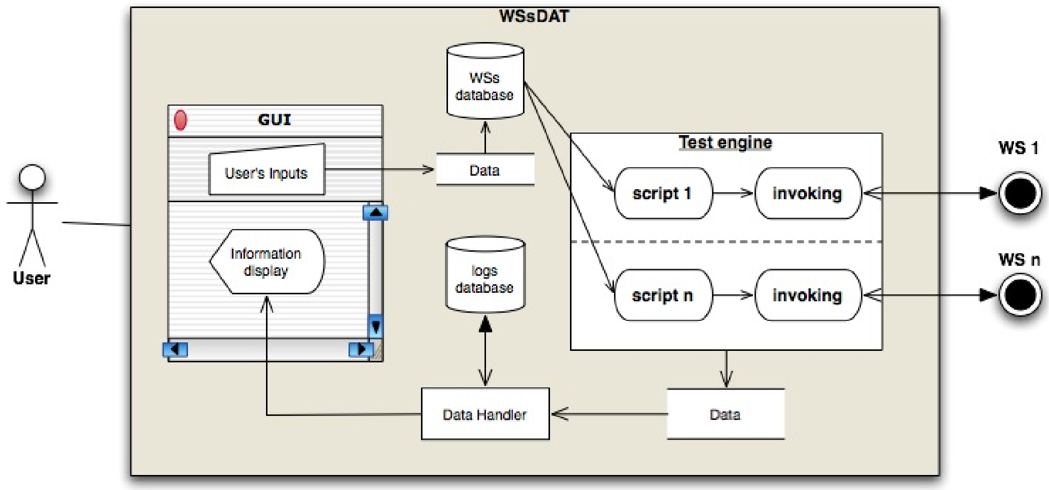


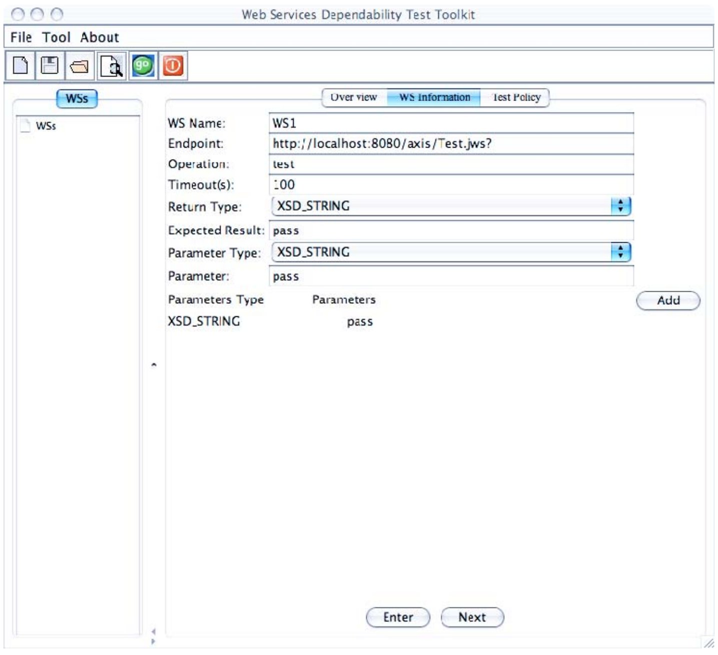
Figure 1: The architecture of the WSsDAT

2.2 General principles and architecture

One of the problems with using public scientific Web Services is that their interfaces differ from one resource to another. Therefore, testers would normally have to write a customized invocation script for each service because of the different interfaces and parameters required. This kind of problems has been fully aware during the development of the WSsDAT. This tool is implemented using Apache Axis JAX-RPC style SOAP processing APIs and based on an extended Model-View-Controller (MVC) pattern. The architecture of WSsDAT is shown in Figure 1. It consisting of three main functional components, Graphic users interface (GUI), Test Engine and Data Handler. The GUI captures user’s request, configuration of test policy and system settings. These inputs are modelled, mapped and stored in a database for repeated use. The GUI is also a viewport which renders live dependability and performance metrics of the Web Services being monitored. The Test Engine is responsible for generating and executing invocation scripts using the modelled data stored in the Web Services database to invoke Web Services. The Test Engine is able to run a batch of tests and measurements concurrently. The Data Handler processes and models all test and observation measurements data. After statistic computation, these data are subsequently stored in a MySQL database or as plain text files; relevant information is passed and rendered in the viewport on the GUI.

2.3 Graphical user interface (GUI)

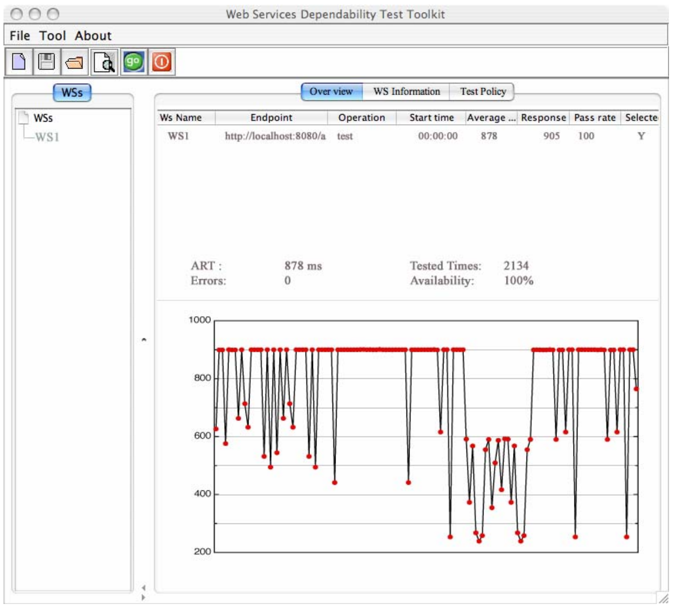
We designed and implemented the GUI by which users can interact with the WSsDAT. Users can input information of Web Services on the GUI, set test parameters and configure test policies, as shown in Figure 2.a. The WSsDAT is capable of testing multiple Web Services simultaneously. Each time the GUI accepts inputs for one Web Service. Once user's inputs are checked to be valid, these data are saved in a database after modelling, and the Web Service is entered into a test array, then the user can input information for the next Web Service. The Web Services in the test array are listed on the GUI and can be selected individually for modification and information display. The viewport on the GUI renders information related to a Web Service, such as errors, average response time, and graph of response times, when user highlighted this Web Service in the list (see Figure 2.b).



WS Name: WS1
Endpoint: http://localhost:8080/axis/Test.jws?
Operation: test
Timeout(s): 100
Return Type: XSD_STRING
Expected Result: pass
Parameter Type: XSD_STRING
Parameter: pass

Parameters Type	Parameters
XSD_STRING	pass

Figure 2.a: GUI for Web Services information inputs



Ws Name	Endpoint	Operation	Start time	Average ...	Response	Pass rate	Selecte
WS1	http://localhost:8080/a	test	00:00:00	878	905	100	Y

ART :	878 ms	Tested Times:	2134
Errors:	0	Availability:	100%

Figure 2.b: GUI for test information display

2.4 Test engine

The Test Engine processes the user's inputs and implements service invocation scripts according to test policies. Test on each Web Service is established as one thread and all tests are carried out in parallel. The number of test threads is only restricted by the computer system's capability or restriction. Figure 3 is an UML diagram shows how the Test Engine cooperates with other components in the WSsDAT. The mechanism of a test procedure described briefly as following:

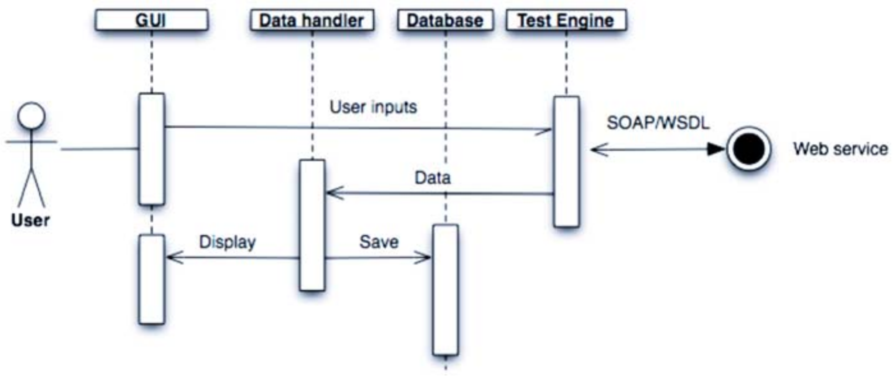


Figure 3: Test procedure

- The Test Engine assembles an invocation script for the service to be monitored according to user's inputs.
- The Test Engine invokes the Web Service with the test script, and a timer is started for measuring the response time. The start time is logged.
- If a valid result is returned from the Web Service, the result is passed to the Data Handler along with other measurements such as start time, end time and round trip time. The test is terminated and will be started again after a preset interval.
- If an exception is detected from the Web Service, the message is logged along with other dependability and performance metrics. The test is terminated and a new invocation will be initiated after a preset interval.
- If the Web Service does not return any response after a preset time out period, the time out exception is logged. The test is terminated and another call to the service will be made after the preset interval.
- When a test is terminated, relevant statistics are processed and data are logged.

2.5 Data handler

The Data Handler processes all data generated during the test. After statistic computation, these data are stored in a MySQL database, and passed to the GUI if relevant. If a MySQL system is not installed on the computer, The WSsDAT has an option to save these data in formatted text files. The contents of these files are commented and split clearly and can be easily converted into Microsoft Excel and some other statistics software which can import data from formatted text files such as Pro-fit and SPSS.

3. Related works

There are some other solutions available on Web Services dependability and performance monitoring. Some types of tools are intended to help Web Service developers to improve the dependability of the service under development. For example, paper Townend *et al*., 2005 describes a fault injection tool that allows the dependability of a service to be assessed by injecting various XML messages which have the correct syntax but incorrect values of parameters. This tool works under the assumption that additional software “hooks” can be inserted into the service before the experiment starts.

Another Web-based monitoring tool has been recently developed by Schmietendorf *et al.*, 2005 for measuring the performance of Web Services. Web Services developers can use this tool to monitor performance measurements of their Web Services, for instance ping time and call time to the Web Service and errors. All tests and measurements are actually carried out from the Website where the tool is located. In order to use this tool, users, who must have administration rights to the Web Service, need to register their Web Services on the Web page of the tool or via its WSDL interface, and provide the URL of the WSDL document of the Web Services. Then the tool processes the documents with some necessary parameters specified by users to dynamically generate test scripts. The tool is able to log measurements results; the format of log file is exchangeable to Excel or SPSS. The tool also has reporting function can show graphical analysis on its website.

There are some Web Services monitoring tools acting as proxies to the Web Services under monitoring. For example, the Bionanny system (Bionanny, 2005), which is actually a Web Service itself. It intercepts the communication between clients and destination Web Services. Therefore it is capable to monitor the usage of the Web Services. The Bioanny also provides APIs to log characteristics about requests and services for further analysis. This tool is developed for Web Services providers. To use the Bioanny, users need to install Java, Apache Tomcat servlet engine, Apache Axis SOAP toolkit and MySQL database on their system, which in effect means that the users need to build a Web Service environment to deploy the Bioanny. Ideally the Bioanny tool needs to be installed in the server where the Web Services to be monitored are located.

Commercial solutions for Web Services monitoring are quite rare nowadays. eValid™ Web Test and Analysis Suite (eVaild, 2005) is one originally developed for Website monitoring. The eVaild™ is a test enabled IE-compatible browser type tool which is mainly intended for monitoring Websites. It is claimed to be capable of monitoring and testing Web Services as long as it can be viewed in a IE-compatible browser, say eVaild™. eValid™ has the following functionalities which can be related to Web Services:

- Functional test: it emulates user's behavior to assure the reliability measurements.
- Loadtest: it measures feedbacks for emulated payloads for more accurate analysis.
- Monitoring: it pings and checks accessibility of a Website/Web Service.
- Performance check: it examines download times of each element from the server; therefore users can analyze the bottle-necks.

Although the eValid™ is not a dedicated tool for Web Services monitoring. It is not able to test and monitor Web Services via invoking their APIs. However it does provide a way to monitor Web Services that has additional Dynamic Webpage accesses.

4. Experiments on using WSsDAT

4.1 BLAST services

BLAST is an algorithm which is commonly used in *silico* experiments in bioinformatics to search for gene and protein sequences that are similar to a given input query sequence (Altschul *et al.*, 1990). BLAST has been deployed on numerous servers around the world but their dependability differs from one service to another. For a computationally-intensive and long running *in silico* experiment, it is important that the most efficient and reliable service is used so that chances of experiment failure is reduced. To this end, the most reliable BLAST service might be used based on dependability and performance characteristics which have been measured by the WSsDAT tool.

4.2 Testing BLAST Web Services

An experiment measuring the performance of two BLAST Web Services was undertaken to test the monitoring capability of the WSsDAT tool. Dependability and performance metrics were measured from a Soaplab BLAST Web Service deployed by the European Bioinformatics Institute, Cambridge, UK (Senger *et al.*, 2004) and a BLAST service hosted by the DNA Databank in Japan (DDBJ) (Miyazaki and Sugawara, 2000).

We deployed the WSsDAT tool at three different locations and networks to monitor these two BLAST Web Services simultaneously. Two WSsDAT tools were located in Newcastle upon Tyne, UK; one was deployed from the campus network at University of Newcastle upon Tyne, whilst the other one was hosted on a computer connected with 1MB broadband via a commercial Internet Service Provider, Telewest Broadband (UK). The remaining WSsDAT was deployed in the China Education and Research Network, an official Internet established by the Chinese government.

In order to observe the variances of dependability and performance metrics at different periods of time in a working day, over the weekend, and during daytime and night time hours, the two BLAST services were monitored from Friday, March 18, 2005 until Sunday, March 20, 2005. The total duration was 72 hours and the interval between successive service invocations was 30 minutes. Since the results returned from the BLAST services can be vary depending on the variables hidden behind each BLAST service interface, we did not check the semantics of the BLAST results. All measurements were stored in a database for further analysis.

4.3 Data analysis and discussion

The response times from the BLAST Web services hosted at the EBI in Cambridge, UK and at the DDBJ were monitored by the WSsDAT tool from three separate locations for a period of 72 hours (see Figure 4). During this period, the response times for successful invocations of the EBI BLAST service varied dramatically from 239 to 760 seconds when invoked from within the

University of Newcastle campus network, 248 to 1000 seconds when invoked from the commercial broadband Internet, and 261 to 1886 seconds when invoked from China (Figure 4).

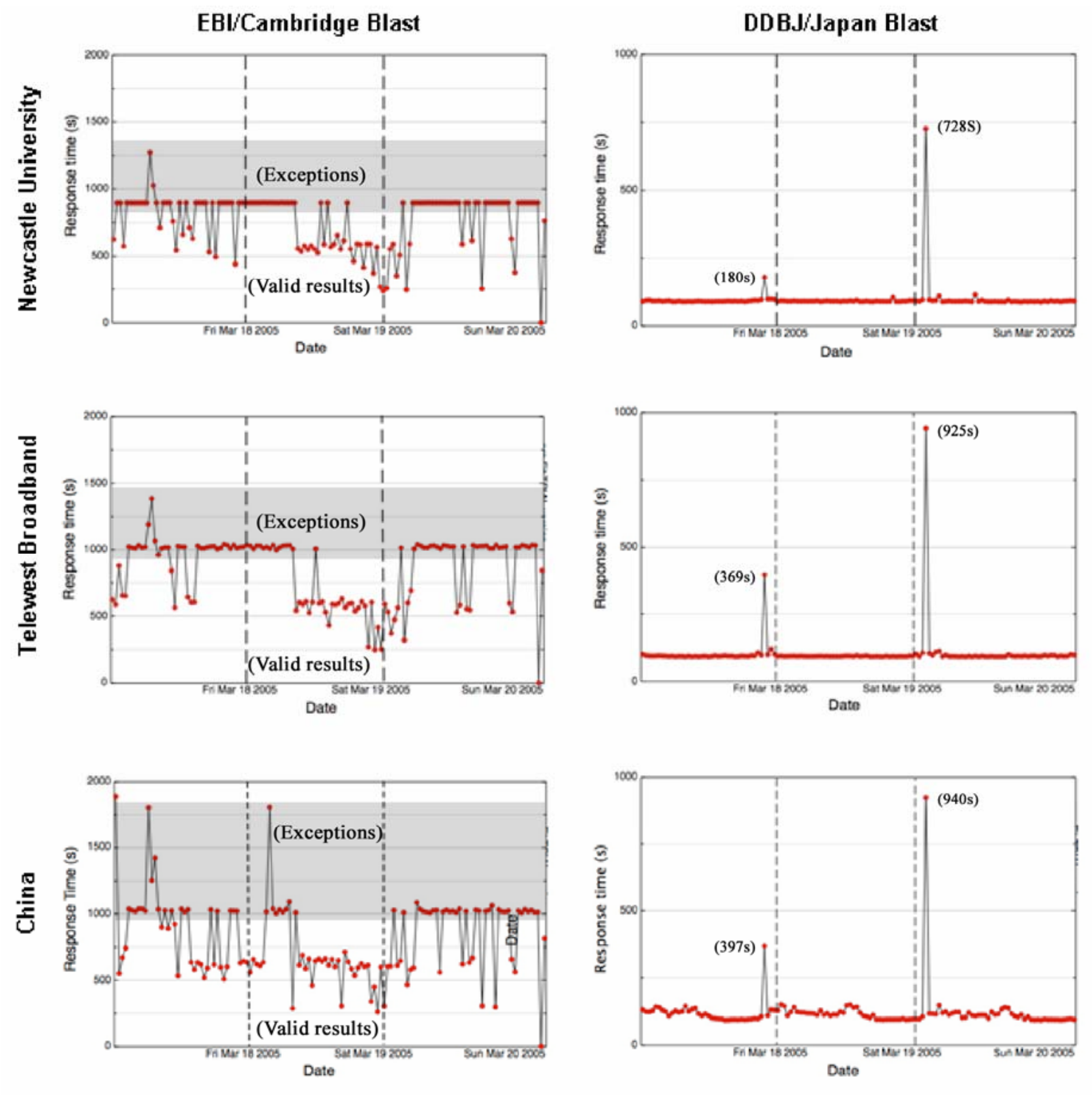


Figure 4: Performance metrics measured using WSsDAT from BLAST services deployed at EBI and DDBJ when invoked from University of Newcastle campus network, a commercial broadband supplier (UK) and from China. Service failures have been shaded as grey.

The graphs of the response time curves also differed according to the location of the invoking WSsDAT client such that the average response times from the EBI BLAST service were quicker when invoked from Newcastle University compared to both the commercial broadband network and in China (Figure 4). It may due to networking consequences, or, sometimes, heavy service load.

The average successful response times from Newcastle University campus, Telewest broadband and China were 531, 599, and 640 seconds, respectively. However, when we focus on the failures happened during the test, the comparisons were much different. In the 72 hours test period, 130 service invocations were executed from each of the three locations. The number of successful tests of the EBI BLAST service was only 47 from Campus, failure rate was 64%; the numbers were 54 and 74 from Telewest Broadband and China, and the failure rates were 58% and 43%, (failure responses are in gray areas of each graph in Figure 4).

All these failures were described with an exception message “Gateway time out” received by the three WSsDAT clients. We do not have adequate information to comment where this time out exactly happened. However, the WSsDAT client in China received a valid result from the EBI BLAST service after 1886 seconds, which was the longest response time during the test. This suggests that the time out failures were not due to the WSsDAT itself. It is interesting to note that service failures rarely happened between Saturday daytime and Sunday early morning (Figure 4).

In contrast, all invocations on the DDBJ BLAST service in Japan succeeded from all the WSsDAT clients (see Figure 4). The response times from all these three locations remained relatively constant with averages of 92, 96 and 114 seconds from Newcastle University campus network, Telewest broadband (UK) and China, respectively, which again suggests that networking conditions can differ between geographical locations. All the three tools simultaneously recorded two slow responses from the DDBJ during the 72 hours test caused by some unknown Web Service problems.

The experiments on the two BLAST Web Services showed that WSsDAT can record real-time information on the dependability and performance metrics of services. The results of our experiment shows that the dependability of one BLAST service can differ dramatically to another. This information can be used to understand the behaviour of such services and thereby allow scientists to select those which are the most reliable for use in their data analysis. In the context of our experiment, the DDBJ BLAST service should be the analysis of choice of the scientist based on its dependability compared with the EBI BLAST service. Furthermore, the ability to deploy and use WSsDAT to invoke services in different physical locations can lead to insights on how the network can affect the dependability and performance of Web Services.

5. Application of the WSsDAT and future work

There are a number of scenarios in which the information recorded by the WSsDAT tool can be used. The tool will initially be used to assess the reliability characteristics of a set of dedicated e-Science Web Services which are employed within an *in silico* experiment. This information may be collected either before the experiment or during periods when the experiments are usually enacted. The measurements recorded by WSsDAT can be processed in such a fashion which may be used to model the behaviour of the service and the models can then be used to predict the reliability of the service. Furthermore, the information could be used to rank conceptually identical services on the basis of reliability.

The information collected and generated by WSsDAT can then be used by the scientist during the construction of the *in silico* workflow experiment. From a list of services which all perform a given analysis, the scientist would select that which was forecasted to be the most dependable. In some situations the scientist can decide against running the experiment if the probability of completing it successfully is not sufficient for his/her point of view. In this case the experiment can be postponed until a more favourable time. Otherwise the scientist can decide to employ fault tolerance measures which can be implemented in several ways. Firstly, the experimental results can help the scientist to choose correct time-outs for accessing Web Services (this is typically a serious problem as most of the times the time-outs are chosen without real grounds). Secondly, the scientist can decide if retries can help in calling specific Web Services and he/she can chose a appropriate number of retries. Thirdly, he/she can decide to run experiment in such a way that after several unsuccessful attempts to call a service the system switches to using an alternative service which has been ranked with a lower but still sufficient reliability than the first choice service.

Further development of the WSsDAT will focus on better integration with applications in existing e-Science environment. Some specific future work are planned as to allow the following functionalities:

- Interactive APIs for integration with workflows. The WSsDAT is capable to monitor real time dependability of several Web Services simultaneously. However, as an independent application, its users can only predicatively choose Web Services by manually consulting the information provided by the tool. In order to achieve better integration with workflows and bring greater benefits to e-scientists, we planned to implement interactive APIs in the WSsDAT to allow automatic access to metadata recorded in the database of the tool; therefore e-scientists can use those APIs in their workflows to achieve dynamic reconfiguration of using more dependable Web Services.
- Monitoring Web Services workflow. At this stage the WSsDAT can monitor a Web Services in each thread. In future it will be able to handle interactive communication with the Web Services and monitor workflows automatically.
- Remote control function. Users will be able to deploy several WSsDAT on different computer systems and remotely control the tools and collect data on one computer.
- Roundtrip/Time-out tracking. At the moment, the WSsDAT can only track the roundtrip time. If a timeout happened, the tool cannot locate where the timeout exactly happened. In future, message travelling track mechanism will be implemented in the WSsDAT to track message travelling time in each stage and layer.

6. Conclusions

Developing tools for the experimental evaluation of Web Services dependability is a very active area of research and development. Both service developers and service integrators need to be able to assess the dependability of the individual services. This requires a range of experimental tools that can provide them with the information crucial for making various technical and business decisions.

Our work aims to help the system integrators which do not have any access to the service code and know only the information provided by its WSDL specification (e.g. found for example in the UDDI registry). It will greatly benefit general users especially the workflow developers. To use the WSsDAT monitors Web Services from users' computer, users are able to collect and save real-time information about the candidate Web Services, so that this statistical data can help them in choosing Web Services which will behave more predictably and more dependably. The WSsDAT can also help Web Services developers to improve their Web Services. The platform-independent characteristics of the WSsDAT allow them to conveniently distribute the tool and monitor their Web Services in different contexts using various strategies. For instance, the developers can monitor their Web Services from different geographic locations.

The WSsDAT can be freely downloaded at http://www.students.ncl.ac.uk/yuhui.chen/

7. Acknowledgments.

Alexander Romanovsky is partially supported by IST FP6 RODIN project. Peter Li acknowledges support from the Biotechnology and Biological Sciences Research Council (BBSRC).

8. References

Altschul, S.F., Gish, W., Miller, W., Myers, E. W., Lipman, D. J. (1990) Basic local alignment search tool. J Mol Biol. 215: 403-410.

eValid, Inc. (2005) eValid™ Web Testing & Analysis Suite. http://www.soft.com/eValid/

Ferguson, D.F., Storey, T., Lovering, B., Shewchuk, J. (2003) Secure, Reliable, Transacted Web Services: Architecture and Composition, MS and IBM Technical Report, http://www-106.ibm.com/developerworks/webservices/library/ws-securtrans

Miyazaki, S., Sugawara, H. (2000) Development of DDBJ-XML and its application to a database of cDNA. Genome Informatics. Universal Academy Press, Inc (Tokyo), pp. 380-381.

Oinn, T., Addis, M., Ferris, J., Marvin, D., Greenwood, M., Carver, T., Senger, M., Glover, K., Wipat, A. and Li, P. (2004) Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics 20: 3045-3054.

Randell, B., Romanovsky, A., Rubira, C. M. F., Stroud, R. J., Wu, Z., Xu, J. (1995) From recovery blocks to concurrent atomic actions. In Predictably Dependable Computing Systems. B. Randell, J.-C. Laprie, H. Kopetz, B. Littlewood (Eds.). Springer. pp. 87-101.

Schmietendorf, A., Dumke, R., Stojanov, S. (2005) Performance Aspects in Web Service-based Integration Solutions, 21st UK Performance Engineering Workshop, Computing Science, University of Newcastle upon Tyne. School of Computing Science Technical Report Series, CS-TR-916, July 2005. 137-152. http://www.staff.ncl.ac.uk/nigel.thomas/UKPEW2005/

Senger M, Niemi, M. (2005) Bionanny - A Web Service monitoring other Web Services, http://bionanny.sourceforge.net/

Stevens, R., Robinson, A., Goble, C. A. (2003) myGrid: Personalised Bioinformatics on the Information Grid. Bioinformatics 19 (Suppl 1): i302-i304

Stevens, R., Tipney, H. J., Wroe, C., Oinn, T., Senger, M., Lord, P., Goble, C. A., Brass, A., Tassabehji, M. (2004) Exploring Williams-Beuren Syndrome Using myGrid. Bioinformatics 20: i303-i310.

Tai, A. T., Avizienis, A., Meyer, J. F. (1993) Evaluation of fault-tolerant software: a performability modelling approach. In Dependable Computing for Critical Applications 3. C. E. Landwehr, B. Randell, L. Simoncini (Eds.). Springer. pp.113-134.

Townend, P., Xu, J., Yang, E., Bennett, K., Charters, S., Holliman, N., Looker, N., Munro, M. (2005) The e-Demand project: A Summary. The Fourth UK e-Science Programme All Hands Meeting (AHM 2005) Nottingham UK. 19-22 September 2005. http://www.allhands.org.uk/