www.ietdl.org

Published in IET Software
Received on 28th February 2011
Revised on 18th May 2011
doi: 10.1049/iet-sen.2011.0030

ISSN 1751-8806

Investigation on performance testing and evaluation of PReWebN: a Java technique for implementing web application

M. Kalita[1] *S. Khanikar*[2] *T. Bezboruah*[1]

[1]*Department of Electronics and Communication Technology, Gauhati University, Guwahati 781014, Assam, India*
[2]*En-Geo Consultancy and Research Centre, Guwahati 781001, Assam, India*
E-mail: zbt@gauhati.ac.in; zbt_gu@yahoo.co.in

Abstract: Performance testing of web application is essential from the perspective of users as well as developers, since it directly reflects the behaviour of the application. As such the authors have developed a prototype research web application based on Java technique to study the performance and to evaluate the technique used for developing the web application. The application is called as PReWebN (prototype research web application in netbeans platform). Load and stress testing have been carried out on PReWebN using Mercury LoadRunner to study the performance, stability, scalability, reliability, efficiency and cost effectiveness of the technique. The performance depends on metrics such as hits/s, response time, throughput, errors/s and transaction summary. These metrics for the application are tested with different stress levels. The statistical testing on the recorded data has been carried out to study the stability and quality of the application. The present study reveals that the web application developed with Java technique is more stable, reliable, scalable and cost effective than its other counterpart such as Microsoft .NET technology. The authors present the architecture, testing procedure, results of performance testing as well as the results of statistical testing on recorded data of PReWebN.

1 Introduction

In its inception, web applications were static and essentially read only hypermedia repositories of information. The applications running nowadays on the web introduced a number of innovative aspects. Two main process related to traditional hypermedia repository are (i) the browsing and (ii) the navigation. With user's demand, these two operations have extended to the data entry, business transactions and communication with remote users. Thereafter, the tight integration among different operation paradigms arises. Thus, a web application performs various operations chosen from the point of user's perspective. It has created the need for a new design paradigm.

In world-wide web (WWW), many web applications create dynamic responses to user requests. The dynamic content creation provides operators with various advantages, such as access to information stored in databases, which provides the ability to personalise web applications according to individual user preferences. This facilitates to deliver a much more interactive user experience than that of static web application.

The overall performance of web application depends on components involved, particularly the clients, networks and the servers. The explosive growth of the web results in heavy demand on servers. The server performance has become a critical issue for improving the quality of service (QoS) on WWW [1]. The QoS offered to the users is affected not only by the network bandwidth, but also by processing power of web server [2]. The Apache is one of such popular web servers available for deploying web applications [3]. It is a popular web server on the Internet, used by more than 60% of existing websites [4].

Among the various design paradigms, the model view controller (MVC) is a well-known design pattern to architect interactive web applications. The key idea of this design pattern is to separate the user interfaces (UIs) from the underlying data representation. The model is that part of the application which contains both the information represented by the view and the logic that changes this information in response to user interaction. The controller is the entry point of the application. Use of MVC makes it easier to develop and maintain a web application, because (a) the web application's view can be drastically changed without changing data structure and business logic (BL) and (b) the web application can easily maintain different interfaces, such as multiple languages or different sets of user permissions. The developers can make partitioning easily by deciding which method will run in server and which will run in the client [5].

As such the responsibilities of developers are manifold. While the applications should be provided with ease to use features, at the same time these applications must also be able to handle large number of concurrent users. In such a situation, as most of the businesses are conducted through web, it is very important and crucial to obtain the techniques involved in developing the web application to be

tested. To perform and execute testing, one must be familiar with the system, the inputs, the way they are combined and the operating environment of the system [6]. In general, the testing is performed in the following four phases: (a) modelling the system's environment, (b) selecting test scenarios, (c) running and evaluating test scenarios and (d) analysing test results.

With the massive growth of web user, it is extremely important to verify and measure the reliability and stability of the web application. One of the good candidates for effective web quality assurance is statistical testing. This technique requires collection of large data sets describing various metrics of the application [7].

There are three models that can be used to analyse the performance of an application, namely (i) the simulation (ii) the analytical model and (iii) measurements of existing system. In the analytical model, a set of algorithm is used to analyse the system. It requires the construction of a mathematical model which represents the real system. It provides a quick insight into the system's performance. However, the analysis method tends to be less accurate when compared with the other two techniques. In the simulation model, a computer program is used to generate the parameters needed to perform the test. The specialised computer program captures only features of interest to the study. Although the model eases the design, development and modification, but it limits the accuracy of the result. The measurement of existing system involves benchmarking the system of interest without simplifying assumptions. This method produces more accurate result when compared with the other two techniques but is less flexible.

In view of the above we have designed, developed, implemented and tested a prototype research web application using Java technique (PReWebN), implementing the JavaBeans (Managed beans) as model, Java Server Pages (JSP) along with Hypertext Markup Language (HTML) as view, FacesServlet as the controller and Apache Tomcat as the web server. The PReWebN has been implemented using NetBeans 6.5.1 Integrated Development Environment (IDE). The use of IDE can increase the efficiency during development, as the plug-ins can alert coders if they introduce errors while writing codes [8]. The application has been tested considering the number of virtual users as the main factor with different stress levels and the results of performance testing; statistical testing as well as detail analysis are presented in this paper.

2 Architecture

The Java language is a natural choice for developing web applications. Its strong security guarantees, concurrency control and widespread deployment in both browsers and servers make it relatively easy to create web applications. The Java server faces (JSF) is a standard UI framework for Java web applications. The JSF follows the MVC design paradigm. The rapid web application development is promoted by easily assembling UI components, plumbing them to the back-end BL components and wiring UI generated component events to server-side event handlers. The JSF is a specification for a component-based web application framework [9, 10].

2.1 MVC architecture

Fig. 1 shows the MVC architecture for JSF. The JSF framework has a very well-defined notion of the model, view and the controller. The model contains the BL or non-UI code. It manages the behaviour and data of the web application domain, responds to request for information from the view and responds to instructions from the controller. In event-driven systems, the model notifies view when the information changes, so that they can react. The view contains all the codes necessary to present a UI to the user. It renders the model in a suitable form such as UI components. The controller is a front-end agent that directly handles the user's requests and dispatched the appropriate view. It receives inputs from the view and instructs the model to perform actions based on the input data [9].

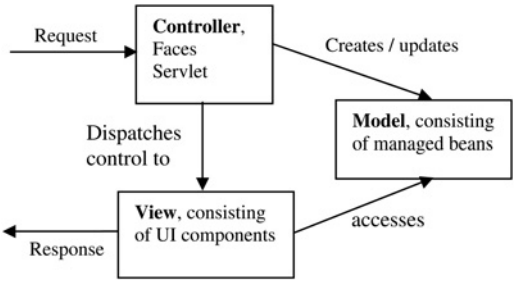

Fig. 1 *Model-view-controller design paradigm*

2.2 Multi-tier architecture of the PReWebN

The PReWebN has been implemented using the visual web JSF to act as the front-end interface to dynamic web content generator. The web content generator is a combination of web server software, the Apache Tomcat and back-end MyStructured Query Language (MySQL) database server. The objective of the experiments is to measure the performance of the front-end dynamic content generator written in JSF with the Apache Tomcat. The architecture of the PReWebN is shown in Fig. 2. The JSF provides a component-centric application programming interface (API) from which web application UIs can be assembled. The JSF specification defines a set of basic UI components that can be extended to achieve more specialised behaviour. The events from client-side UI controls are dispatched to JavaBeans models which provide server-side application behaviour. In JSF, the UI components are loosely coupled to server-side Java plain old Java objects (POJO) which is declared as managed beans. The front-end controller servlet handles all faces requests and dispatches them with the necessary application data to the appropriate view. The database manages the physical storage and retrieval of data. It receives the data from the model and sends it to the database and vice versa. The database queries have been written to access the data from the database and to perform operations like insert, update and delete. In our design, the MySQL acts as the data layer (DL).

3 Design aspect of PReWebN

A web application's design is essentially its look and feel [11]. We have taken into account all the web elements, for example, audience information, purpose and objective statement, domain information, web specification and combine them to produce an arrangement for implementing PReWebN.

The application is developed by considering the profile of the Department of Electronics and Communication

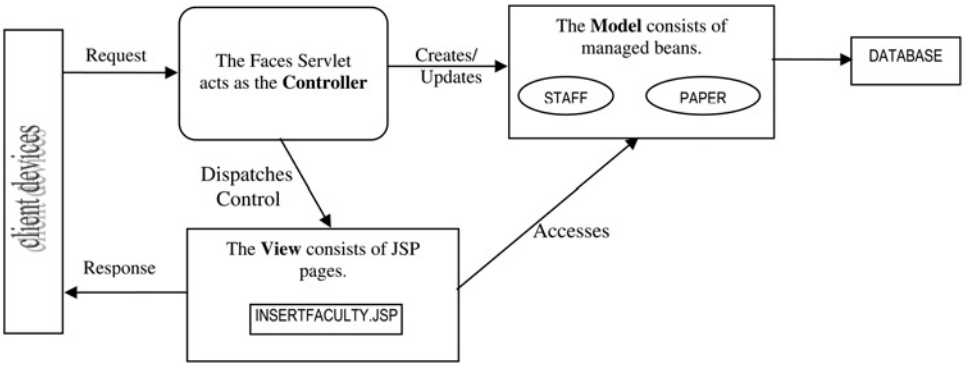

Fig. 2 *Architecture of the PReWebN*

Technology (ECT), Gauhati University as sample data. The Create, Read, Update, Delete (CRUD) operations are performed to generate the response. The basic working principle of PReWebN is shown in the flowchart of Fig. 3. The flowchart is self-explanatory. When users open the web application, the main menu will open. The users can then click any link provided in the main menu. They can search the site or can perform some transactions. For searching, any user can open the site and perform the necessary search. For other operations, for example, insert, modify or delete, users would have to go through the registration process, which is supervised by the administrator. A registered user is authenticated to obtain access to pages performing database transaction. Users can log out at the

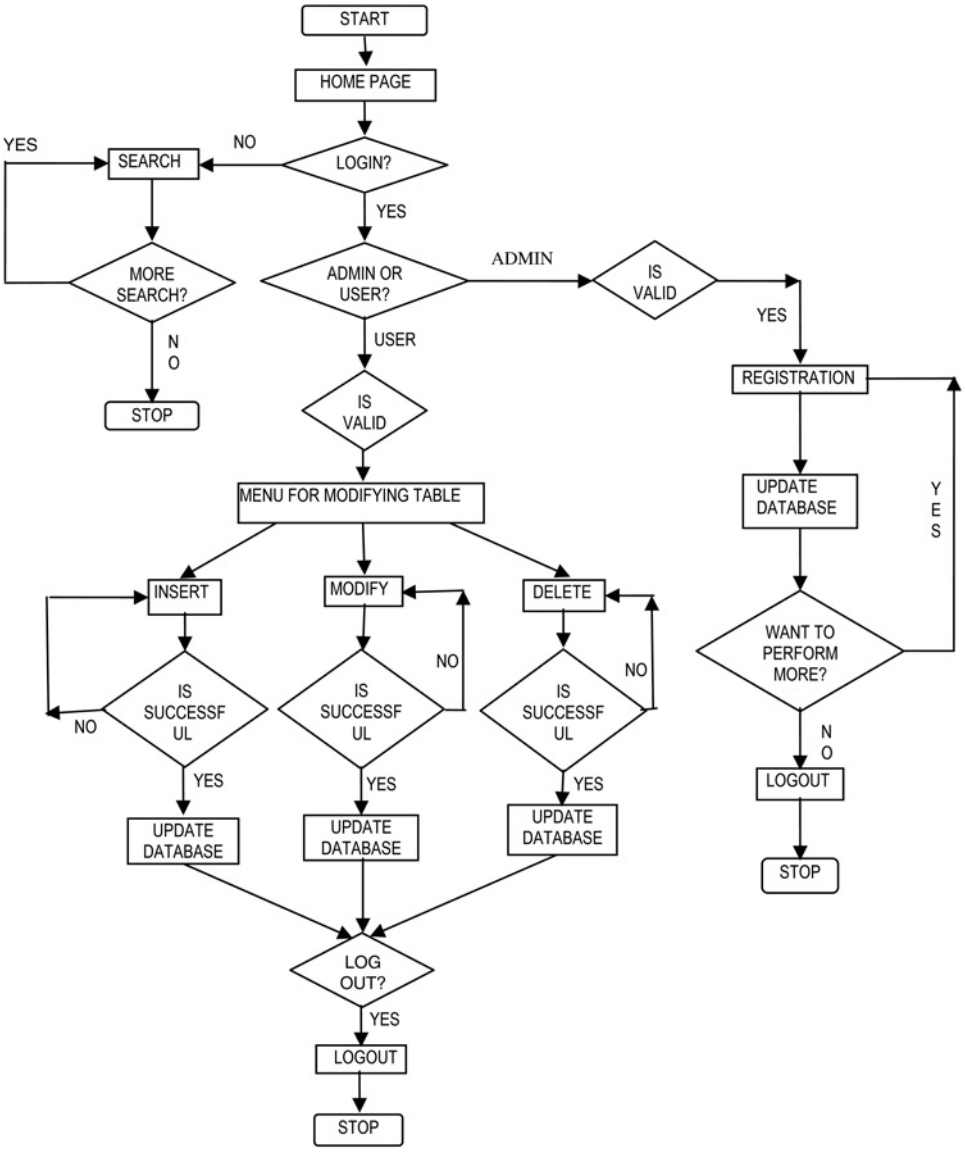

Fig. 3 *Flowchart for basic working principle of PReWebN*

doi: 10.1049/iet-sen.2011.0030

end of his session. The session will be automatically closed if users do not perform any transaction for a specific time period set in the web application. The different module of the system design of PReWebN is shown in Fig. 3.

4 Technical specifications of hardware and software

The detail technical specification of the hardware and the software for the development as well as testing environment for PReWebN is as given below.

4.1 Hardware specification

PC: Intel® Pentium® CPU E2200
Processor speed: @ 2.20 GHz
RAM: 1 GB
Memory space: 150 GB

4.2 Software specification

Web server: Apache Tomcat 6.0.18
Data base server: MySQL 5.0
Browser: Mozilla Firefox
Operating system: Windows XP
Front-end tool: NetBeans 6.5.1

The testbed configuration is shown in Fig. 4.

The web server, database server and the workload generator are run in the same machine. The web server, the database server and the workload generator (client emulator software) run on an Intel Pentium CPU E2200 machine with processor speed of 2.20 GHz. The network bandwidth is chosen to be of 128 kbps considering the fact that if this bandwidth is supported by the system in a decent manner then higher bandwidths are always acceptable.

5 Testing of PReWebN

5.1 Testing approaches

The software testing is a process to evaluate efficiency, performance, scalability, portability, compliance, interoperability and effectiveness of a system. In software development, testing is used at key checkpoints in an overall process to determine whether objectives are being met or not. The code of the web application is tested as unit or module level by the programmer. At the system level, the developer or independent reviewer may subject a service to one or more performance tests [12]. A web application's QoS is measured in terms of response time, throughput and availability. One of the best ways to measure an application's QoS is to conduct load testing.

The performance testing is a testing that is performed, from one perspective, to determine how fast some aspects of a system perform under a particular workload. It can serve to validate and verify scalability, reliability and usage of resources. It can also demonstrate whether the system meets

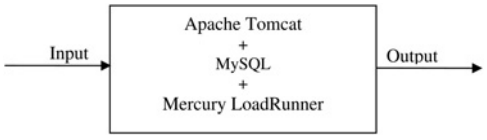

Fig. 4 *Diagrammatic representation of the testbed configuration*

performance criteria or not. It can also diagnose the part of software that contributes most to the poor performance of the system [13, 14].

After verifying the correctness of the code, the load and stress testing is performed to measure the performance and scalability of the application under heavy load. After analysing the results obtained during this phase, it is possible to determine the bottlenecks, memory leakage or performance problems related to the DL.

The load testing is done to have an overall insight of the system. It models the behaviour of users in real world. The load generator mimics browser behaviour and each emulated browser is called a virtual user [15]. In load testing the system is subjected through reasonable load in terms of number of virtual users to find out the performance of the system, mainly in terms of response time. In this case the load is varied from zero to the maximum up to which the system can handle in a decent manner.

The stress testing is performed to determine the stability of a given system. It involves testing beyond the normal operational capacity. The stress testing is performed to uncover memory leakage, bandwidth limits, transactional problems, resource locking, hardware limitations and synchronisation problems that occur when an application is loaded beyond the limits determined by the performance statistics [16–18].

5.2 Testing of PReWebN

The Mercury LoadRunner (version 8.0) is used for testing of PReWebN. It is an automated performance and load testing tool for studying system behaviour and performance, while generating actual load [19]. The testing is carried out to determine the responsiveness, throughput, reliability and scalability of the combination of JSF with Apache Tomcat under a given workload. The operations performed are insertion, updation, deletion and search. The testing is object based in a sense that the whole application has been divided into different objects [20].

During the experiments, the stress level is gradually varied, so that it can saturate the server. This can lead to find out the capability of the server. Each HTTP requests causes a standard query language (SQL) INSERT commands to insert two text fields in the database table. After invoking the application, users will log onto the PReWebN using a unique username and password. Successful login will authenticate users to perform the transaction. Real-life values are inserted into the text fields. The values can then be saved into the corresponding database table by pressing the 'SAVE' button. User think time of approximately 10 s is incorporated in performing each transaction. The run time for experiments was chosen subject to two constraints: (i) the test duration must be long enough to assess accurately the server's ability to support the target request rate and (ii) the duration of each test should be as short as possible, in order to test many request and accurately identify the peak performance of each server. Considering the above facts, an average steady-state period of 30 min is fixed for all the experiments. The test case is tabulated in Table 1.

A flow diagram of different steps involved in the testing of PReWebN is shown in Fig. 5. The different steps involved in performing the performance test include the following subsections.

5.2.1 Preparation of test plan: Various test cases are designed in this step to view performance test from user's

Table 1 Test case

Test case: Insert record				
Description: This test case inserts a person record in a specified database table				
Data requirements:				
{username} – User must have insert privileges. User names must be unique {password} – Must be valid for a given username				
Step number	Step description	Expected result	Transaction name	User think time (s)
1	invoke application. Log in using username and password. Press 'LOGIN' button	user is authenticated to perform insert operation	Init_Transaction	10
2	enter User ID and User name and press button 'SAVE'	the values are inserted into database table	Action_Transaction	10
3	log out pressing button 'LOGOUT'	user is logged out and redirected to the home page	End_Transaction	10

perspective which includes architectures of client PC, browsers and network bandwidth.

5.2.2 Creation of test environment: The test environment consists of hardware and software configuration of client PC, operating system and number of client PC participating in the test.

5.2.3 Set stress level: The number of virtual users participating in the test is chosen during this step.

5.2.4 Create new scenario: The new scenario is created in this step where the period for which the stress test will be performed is decided. There may be three phases: (a) the ramp up phase initialises the system until it reaches a steady state, (b) the steady-state phase where the measurements are taken and (c) the ramp down phase which allows the PReWebN to cool down. User think time and the operation to be performed are also decided.

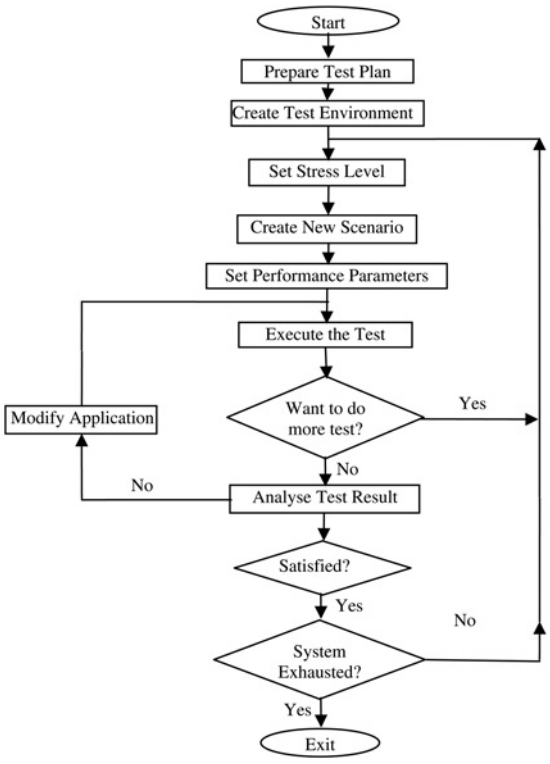

Fig. 5 *Flowchart for testing procedure*

5.2.5 Setting of performance parameters: The performance parameters are important to measure the performance of the application. All the parameters are not relevant to us. The parameters which are important to our present work are set, for example, stress level and bandwidth are set during this step.

5.2.6 Execution of the test: After setting the performance parameters the test is executed by deploying the applications on Mercury LoadRunner.

5.2.7 Analysis of test results: The test responses which are in the forms of graphs are then analysed to evaluate the performance of the application developed by using Java technique in this step.

6 Performance parameters and metrics of PReWebN

The performance has been analysed from user's as well as developer's point of view. The users are more concerned with no refusal of connection having fast response at the client end. On the other hand, at the server end, the developer is concerned with high connection throughput and high availability. The factors on which the web server performance depends are (i) the hardware and software platform, (ii) the operating system, (iii) the server software, (iv) the network bandwidth and (v) the load on the server.

6.1 Testing parameters

There are three main parameters which are varied during the testing procedure. They are (a) the workload intensity measured in terms of number of virtual users, that is, stress level, (b) the workload mix which defines what users will do in each session and (c) the user behaviour parameter, which is the think time.

6.2 Test responses

The metrics of the load and stress test which we have monitored include (i) the response time in second, (ii) the

doi: 10.1049/iet-sen.2011.0030

Table 2 Virtual user level results

Scenario	No. of users	Recorded metrics	Average	Connection refusal in %
insert operation	30	response time (s)	47.6	0
		throughput (bytes/s)	98 806	
		hits/s	2.248	
	75	response time (s)	56.6	26
		throughput (bytes/s)	129 440	
		hits/s	3.681	
	100	response time (s)	112.34	50
		throughput (bytes/s)	122 645	
		hits/s	2.524	
	125	response time (s)	60.85	62
		throughput (bytes/s)	103 503	
		hits/s	2.6	

throughput in bytes per second, (iii) the hits per second, (iv) the number of successful virtual users allowed for transaction, (v) the transaction summary which includes the number of completed and abandoned sessions and (vi) the error report.

6.3 Experimental results

The performance testing is carried out for 10, 20, 30, 40, 50, 75, 100 and 125 virtual users. All performance metrics are measured in 128 Kilobytes/s bandwidth for the reason as discussed in Section 4.2. We observed various metrics provided by the LoadRunner. It is seen that up to 50 users the application runs smoothly. All the performance tests are conducted with ramp up schedule of 30 s. They are phased out at the same time after the completion of the steady-state period [21]. The delays for the users think time is included to emulate the behaviour of real users. User think time included is 10 s. The virtual user levels 75, 100 and 125 are tested to force the web application to work beyond its capacity. The results are given in Table 2.

Some sample responses of the tested metrics are shown in Figs. 6–8, respectively. Fig. 6 shows the response for hits/s against the number of users for 75 virtual users. In this case 56 users are allowed to perform the transaction, rest are failed. It is observed that hits/s increases with number of virtual users, becomes maximum at around 30 virtual users and then decreases gradually. The recorded average hits/s is 3.681.

Fig. 7 shows the response for throughput against number of users for 75 virtual users. It is observed that the throughput increases with number of virtual users, becomes maximum at around 50 virtual users and then decreases gradually. The recorded average throughput is 129440.023 bytes per second.

Fig. 8 shows the response for response time against number of users for 75 virtual users. It is observed that the response time increases initially with number of virtual users, then it reached a steady state and then shoots to a maximum level with increasing virtual users at around 51 users. The recorded average response time is 56.69 s.

7 Statistical analysis of PReWebN

The statistical analysis of PReWebN is performed for 10 virtual users run for 5 min in steady state. The user think time incorporated here is 70 s. The same set of test scenario was repeated for 30 times which is the number of samples. The observed metrics are given in tables below to analyse for evaluating the reliability and stability of the application.

7.1 Analysis of distribution for response time, hits/s and throughputs with equal bin size

The difference between best case and worst case in the performance metrics are divided into seven bins of equal width according to our convenience. The class interval and frequency for response time, throughput and hits/s are shown in Tables 3–5, respectively. Our objective is to determine the distribution of response time, hits/s and throughputs. One of the ways of determination is to plot a

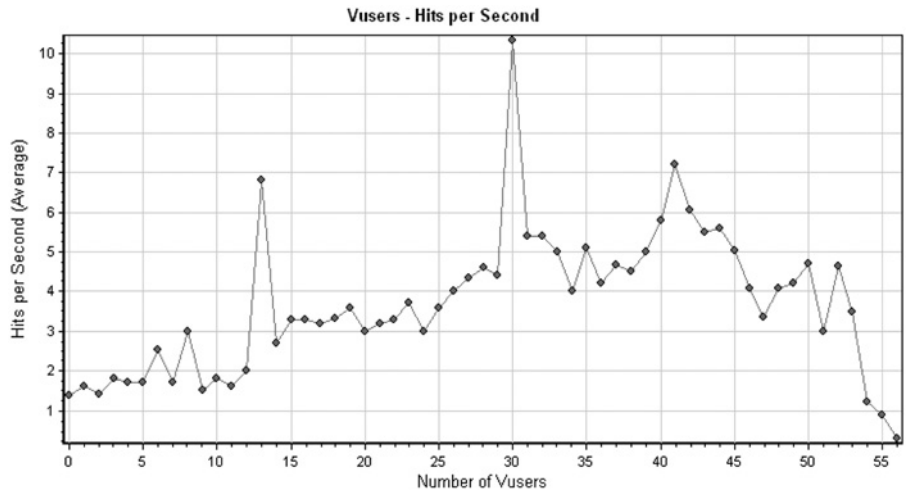

Fig. 6 *Hits/s against number of users for 75 virtual users*

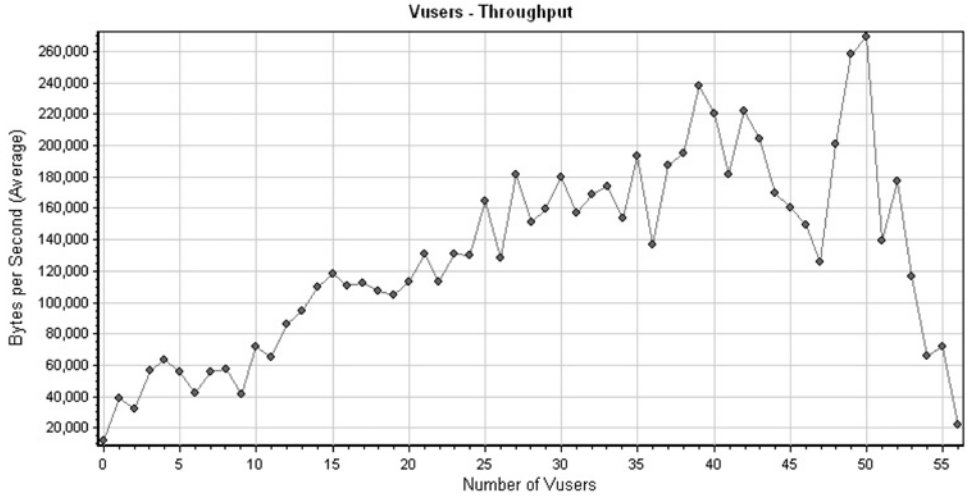

Fig. 7 *Throughput against number of users for 75 virtual users*

Fig. 8 *Response time against number of users for 75 virtual users*

histogram of the observed metrics as shown in Figs. 9*a*–*c*, respectively. The applied distribution is normal distribution according to the histograms. But there is a major drawback with histograms, that is, depending on the used bin sizes; it is possible to draw very different conclusions.

A better technique is to plot the observed quantiles against the recorded data in a quantile plot [22]. If the distribution of observed data is normal, the plot is close to be linear. The resultant plots are shown in Figs. 10*a*–*c*. Based on the observed data, the response time, throughput and hits/s do appear to be normally distributed.

The test of normality can be verified graphically, using the normal probability plot. If the data samples are taken from a normal distribution, the plot will appear to be linear. The normal probability plots of the response time, throughput and hits/s are shown in Figs. 11*a*–*c*. The data follow a straight line, which predicts that the distribution is a normal one.

It is difficult to manage the size of the bins in a suitable manner. If they are big enough, they might smooth out useful information. To avoid this problem cumulative distribution function (CDF) of the response time is calculated and plotted. Fig. 12 represents the plot for CDF of response time which shows the distribution to be continuous one.

Table 3 Class interval and frequency for response time

Response time, s	Observed frequency
118.269–121.269	4
>121.269–124.269	4
>124.269–127.269	4
>127.269–130.269	5
>130.269–133.269	7
>133.269–136.269	4
>136.269–139.269	2

Table 4 Class interval and frequency for throughput

Throughput, bytes/s	Observed frequency
60 552–61 817	4
>61 817–63 082	5
>63 082–64 347	7
>64 347–65 512	6
>65 512–66 777	3
>66 777–68 042	3
>68 042	2

7.2 Analysis of distribution for response time, hits/s and throughputs with unequal bin size

The same set of data is also analysed considering unequal bin size for the histograms. We assume the minimum interval as 3 while the maximum interval is 6 (3×2, i.e. twice the minimum value). The minimum interval is taken as the base interval and the others are based on it. Hence the frequency in the range 127.269–133.269 is divided by 2, which gives 6. Similarly, the frequencies for hits/s and throughput are scaled based on the class interval. The class interval and frequency

Table 5 Class interval and frequency for hits/s

Hits/s	Observed frequency
1–1.014	3
>1.014–1.028	3
>1.028–1.042	3
>1.042–1.056	8
>1.056–1.068	6
>1.068–1.082	4
>1.082–1.097	3

doi: 10.1049/iet-sen.2011.0030

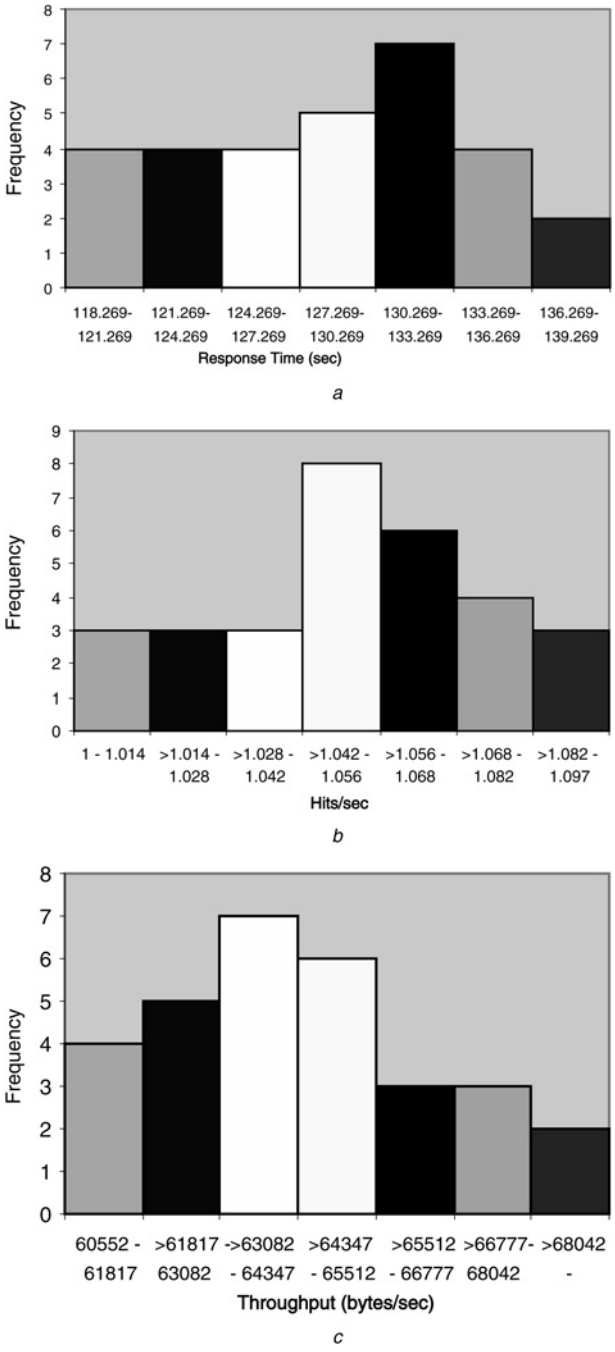

Fig. 9 *Plots of histogram of observed metrics*

a Histogram for response time
b Histogram for hits/s
c Histogram for throughput

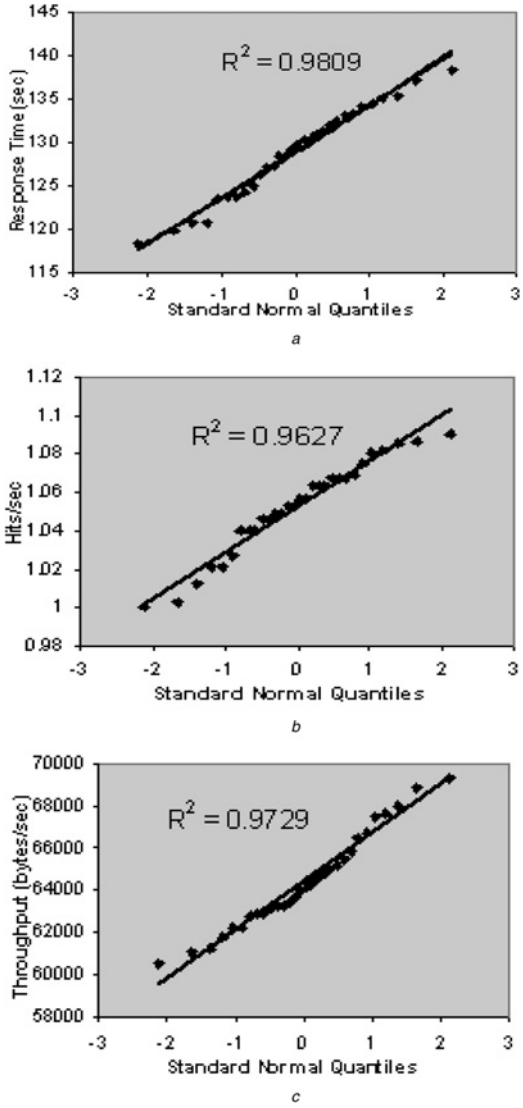

Fig. 10 *Resultant quantile plots*

a Quantile plot for response time
b Quantile plot for hits/s
c Quantile plot for throughput

for response time, throughput and hits/s are given in Tables 6–8, respectively. The histograms for response time, hits/s and throughput are shown in Figs. 13*a*–*c*, respectively. As in case of considering equal bin size, the applied distribution is a normal one when we consider unequal bin size that can be observed from the histograms shown in Figs. 13*a*–*c*.

7.3 Confidence interval of response time, hits/s and throughput

The 95% confidence interval for the mean values of response time, hits/s and throughputs are estimated. The population means μ can be expressed as [23, 24]

$$\mu = \bar{x} \pm \frac{t_c S}{\sqrt{N}} \tag{1}$$

where $\bar{x}$ = mean value, $t_{c(0.05,29)}$ = critical value, S = standard deviation, N = no. of samples and $(t_c S/\sqrt{N})$ = margin of errors.

Considering different values of metrics, obtained during load testing, we evaluate the critical value, mean and margin of errors which are tabulated in Table 9. The population mean μ is calculated from (1). From Table 9, we can conclude with 95% confidence, for 10 virtual users, the range for mean response time is between 128.7858 ± 2, that is, 126.8–130.78 s, mean hits/s is between 1.043 and 1.06, and mean throughput is between 63 543 and 65 277 bytes/s.

7.4 Factors influencing the response time

To verify whether there is a relationship between response time, hits/s and throughput, we assume that such a relation, if exists, be a linear one. The response time is assumed as

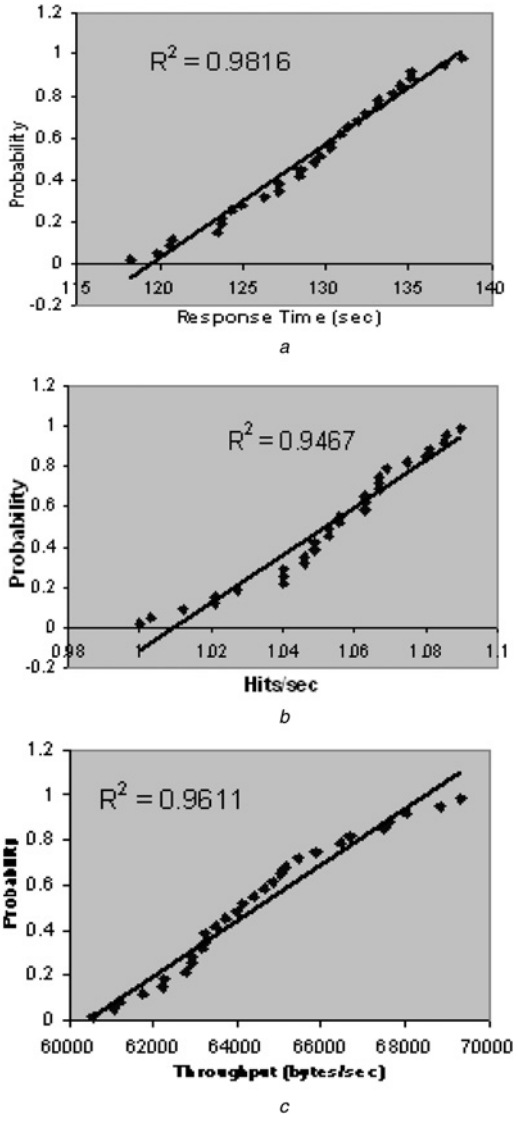

Fig. 11 *Normal probability plots*

a Normal probability plot for response time
b Normal probability plot for hits/s
c Normal probability plot for throughput

criterion variable. The criterion variable is the variable being predicted that depends on values of other variables. Hits/s and throughput are assumed to be predictor variables. The predictor variable is a variable that can be used to predict the value of another variable. The scatter plots of the

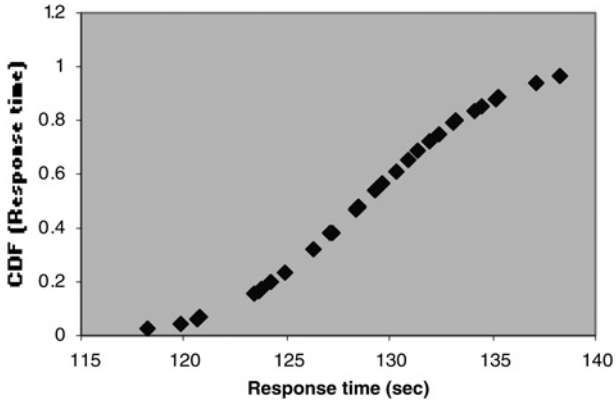

Fig. 12 *CDF of response time*

Table 6 Class interval and frequency for response time

Response time, s	Observed frequency
118.269–121.269	4
>121.269–124.269	4
>124.269–127.269	4
>127.269–133.269	12
>133.269–126.269	4
>136.269–139.269	2

Table 7 Class interval and frequency for throughput

Throughput, bytes/s	Observed frequency
60 552–61 817	4
>61 817–63 082	5
>63 082–65 512	13
>65 512–68 042	6
>68 042	2

Table 8 Class interval and frequency for hits/s

Hits/s	Observed frequency
1–1.014	3
>1.014–1.042	6
>1.042–1.068	14
>1.068–1.097	7

response time against hit/s and against throughput are given in Figs. 14*a* and *b*.

The two scatter plots with their respective regression line shows linear relationship. Greater the value of hits/s, more will be the response time. In a similar way, the greater the value of throughput more will be the response time. To examine the combined effect of throughput and hits/s on response time, we performed multiple linear regression tests. The test is carried out at 95% confidence level. We assumed the null hypothesis (H_0) which is response time does not depend on hits/s and throughput. The alternate hypothesis (H_1) is response time is a function of hits/s and throughput.

The multiple linear regression analysis is carried out using MS Excel. The analysis of variance (ANOVA) provides F ratio to be 5.416881, which is greater than the critical value. The critical value from the F table at significance level 0.05 is ($F_{2,27}$) 3.36, where 2 and 27 are regression and residual, respectively. This concludes F ratio to be significant at 0.05. This provides evidence of existence of linear relationship between response time, hits/s and throughput. As such, the null hypothesis may be rejected. This implies that the equation has 95% chance of being true. The analysis also suggests that our model accounts for 28.63% variance on response time. Thus, we may infer that the hits/s and throughput have some influence on response time.

8 Discussion and conclusion

The objective of our present investigation is to evaluate the overall performance of the Java technique for developing

doi: 10.1049/iet-sen.2011.0030

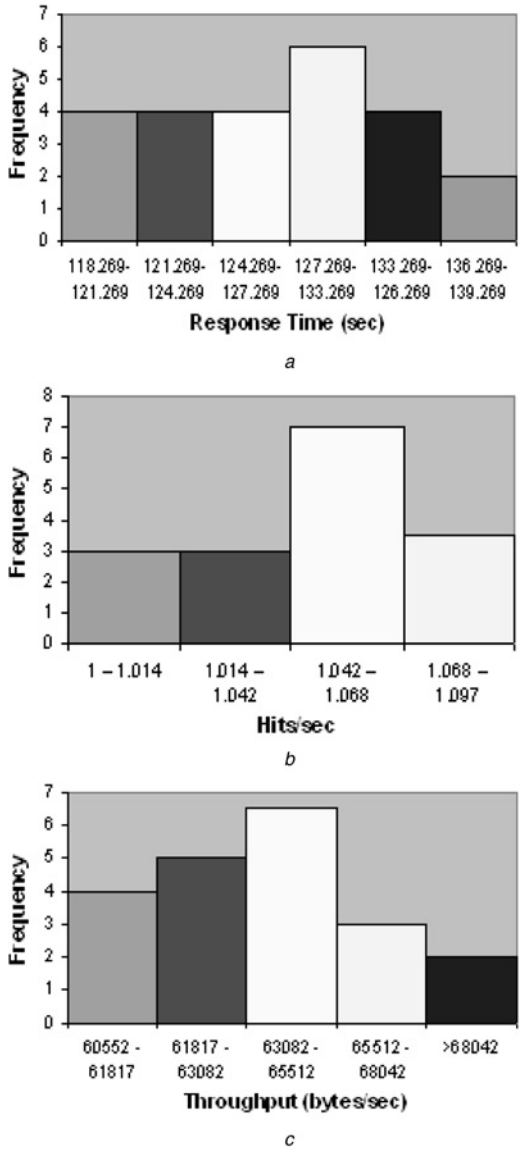

Fig. 13 *Different histograms with unequal bin size*

a Histogram for response time with unequal bin size
b Histogram for hits/s with unequal bin size
c Histogram for throughput with unequal bin size

web application. The analysis of the recorded data predicts that up to 50 virtual users the application developed with Java technique (PReWebN) shows ideal response without any refusal in connectivity with an average response time of 96.72 s. As we increase the number of virtual users, the errors per page of the application as well as connectivity errors increase. For 40 virtual users the average response time is 42.17 s having no refusal of connection. Similarly for 75 virtual users the average response time is 56.6 s and 26% connection is refused. For 100 virtual users the average response time is 112.34 s with 50% connection

Table 9 Different values of metrics

N	$T_{0.05,\ 29}$	Metrics	$(\bar{x})$	S	$\frac{t_c S}{\sqrt{N}}$
30	2.045	response time (s)	128.7858	5.34267	2
		hits/s	1.052	0.024	0.009
		throughput (bytes/s)	64410	2322	867

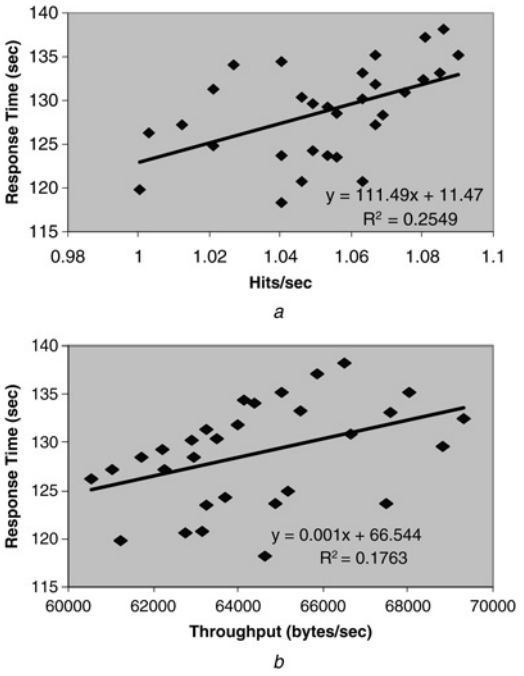

Fig. 14 *Scatter plots of the response time against hit/s and against throughput*

a Hits/s against response time
b Throughput against response time

refusal. Finally for 125 virtual users the average response time is 60.85 s with 62% connection refusal.

The histograms, quantile plots and normal probability plots of the application show linearity and normality, which shows enough evidence for the scalability and reliability of PReWebN with large number of virtual users. However, in some plots histograms are right or left skew. Also, the normal probability plots are not always perfectly straight line but depart from it at the ends. This shows the evidence of longer tails than the normal distribution. From the statistical analysis, it is observed that hits/s and throughput have individual as well as combined effect on response time. Individually, hits/s and throughput contribute approximately 25 and 17% to the response time and together they contribute around 28%.

From the above study and statistics we can conclude that for PReWebN, as we increase the number of virtual users the errors as well as connection refusal increases. The application gets saturated at 75 virtual users. The application almost becomes inoperable near 125 users. The PReWebN implemented using Java techniques seems to be stable up to 50 users with no errors. With the increase in stress level the collisions between requests may increase and hence may less number of hit/s and throughput with increased connection error. Although the application implemented with Java technique complies with industry standard, however, it is essential to test and analyse the system thoroughly to have an in-depth idea of the factors hampering the application. It is also found that the hits/s and throughput have some influence on response time of the application, however is small. The multiple regressions test on the application shows that hits/s and throughput has 28% combined effect on response time. As such, we can conclude that the application implemented with Java technique is reliable and stable considering intermediate number of users with the above-mentioned technical specification.

The refusal of connection at higher number of virtual users may be due to garbage collected heap due to improper release of memory. This may also be due to decrease of response at server end due to increase in number of virtual users. The sudden rise and fall of response time, throughput and hits/s in various output responses may be due to unreleased or lately released of server resources including memory for the consecutive requests. This conclusion however requires thorough and close monitoring and analysis of the server-side resources. This situation occurs more often with higher number of virtual users when the server is stressed due to increased load.

From our overall investigations, it can be concluded that if we consider scalability, stability, reliability and cost, then the Java technique is better than its other counterpart such as Microsoft's .NET technology [25].

9 Future work

The PReWebN may be implemented with large data sets to more accurately benchmark the technique. Recording replica of data for each test and analysing those data may also provide better insight of the technique. Rigorous testing and analysis of PReWebN is necessary to have an in-depth idea of the factors hampering the web application developed with Java technique. A suitable method should be implemented to remove the garbage collected heap during the testing of the application, which may also help for accurate benchmarking of the technique. As part of our future work, we propose to find out the reasons behind the errors and large number of connection refusal for higher number of virtual users.

We also propose to carry out detailed load and stress testing on PReWebN by deploying the web server, application server and workloads in different machines to compare the various responses with the present results.

10 Acknowledgments

This work is supported partially by the University Grant Commission (UGC), Govt. of India. The authors are thankful to the Head, Department of Electronics and Communication Technology, (ECT), Dean Faculty of Technology, Gauhati University for infrastructural support towards the work. The authors are also thankful to Prof. (Mrs.) K. Boruah, Professor and Head Department of Physics; Prof. H.K. Boruah, Professor, Department of Statistics and Formerly Dean, Faculty of Science, Gauhati University for their valuable suggestions during the statistical analysis of the data. The authors are also grateful to the anonymous reviewers for their review and constructive suggestions in shaping the manuscript in final form.

11 References

1 Almeida, M., Almeida, V., Yates, D.J.: 'Measuring the behavior of a world wide web server'. Seventh IFIP Conf. on High Performance Networking (HPN), White Plains, NY, April 1997, pp. 57–72
2 Fujita, Y., Murata, M., Miyahara, H.: 'Performance modeling and evaluation of web systems'. Proc. 1998 IEEE Communication Quality and Reliability Workshop, May 1998
3 Hu, Y., Nanda, A., Yang, Q.: 'Measurement, analysis and performance improvement of the apache web server'. 18th IEEE Int. Performance, Computing and Communications Conf., (IPCCC'99), February 1999
4 Titchkosky, L., Arlitt, M., Williamson, C.: 'A performance comparison of dynamic web technologies', *ACM SIGMETRICS Perform. Eval. Rev.*, 2003, **31**, (3), pp. 2–11
5 Leff, A., Rayfield, J.T.: 'Web-application development using the model/view/controller design pattern'. Fifth IEEE Int. Enterprise Distributed Object Computing Conf., Seattle, Washington, September 2004–2007
6 Whittaker, J.A.: 'What is software testing? And why is it so hard?' (IEEE Software, January/February 2000)
7 Kallepalli, C., Tian, J.: 'Measuring and modeling usage and reliability for statistical web testing', *IEEE Trans. Softw. Eng.*, 2001, **27**, (11), pp. 1023–1036
8 Securing web application across the software development life cycle. White paper published by IBM (2010): http://www.findwhitepapers.com/technology/software_development/software_testing
9 Burns, E., Schalk, C., Griffin, N.: 'JavaServer Faces 2.0: the complete reference' (McGraw-Hill, 2010)
10 Dudney, B., Lehr, J., Willis, B., LeRoy, M.: 'Mastering java server faces' (Wiley Publishing, Inc., 2004)
11 Kalita, M., Bezboruah, T.: 'On HTML and XML based web design and implementation techniques', *Far East J. Electron. Commun.*, 2007, **1**, (1), pp. 65–79
12 What is web testing, http://searchwindevelopment.techtarget.com/sDefinition/0,sid8_gci534970,00.html
13 Software performance testing, http://en.wikipedia.org/wiki/Software_performance_testing
14 Application Performance Testing. White paper, .NET Research Library (2011): http://researchlibrary.theserverside.net/detail/RES/1300815134_825.htmlt
15 Menascé, D.A.: 'Load testing of websites', *IEEE Internet Comput.*, 2002, **6**, (4), pp. 70–74
16 Stress testing, http://www.manageengine.com/products/qengine/stress-testing.html
17 Stress testing, http://en.wikipedia.org/wiki/Stress_testing
18 What's the difference between load and stress testing?, http://www.faqs.org/faqs/software-eng/testing-faq/section-15.html
19 Application-testing tool: Mercury LoadRunner 8.0, http://pcquest.ciol.com/content/software/2004/104093002.asp
20 Subraya, B.M., Subrahmanya, S.V.: 'Object driven performance testing of web applications'. First Asia-Pacific Conf. on Quality Software (APAQS'00), Hong Kong, China, 30–31 October
21 Cecchet, E., Chanda, A., Elnikety, S., Marguerite, J., Zwaenepoel, W.: 'Performance comparison of middleware architectures for generating dynamic web content' (Springer, New York, 2003), pp. 242–261
22 Bogárdi-Mészöly, Á., Szitás, Z., Levendovszky, T., Charaf, H.: 'Investigating factors influencing the response time in ASP.NET web applications', (*LNCS*, **3746**), 2004, pp. 223–233
23 Spiegel, M.R.: 'Theory and problems of probability and statistics' (McGraw-Hill Book Company), SI Edition, Schaum's Outline Series, 2000
24 Pal, S.K.: 'Statistics for geo scientists, techniques and applications' (Concept Publishing Company, New Delhi, 1998)
25 Kalita, M., Bezboruah, T.: 'Investigation on performance testing and evaluation of PReWebD: A. NET technique for implementing web application' (IET Software, 2011), in press

doi: 10.1049/iet-sen.2011.0030