An Investigation into Performance Testing for e-Commerce Web-based Applications

by

Michael J. Mhlabane, BCom(Hons)(IS)

submitted in fulfillment of the requirements for the degree

Master of Technology
in
Information Technology

in the

Department of Information & Communication Technology
Faculty of Commerce

at the

Durban Institute of Technology

Durban, South Africa
January, 2005

Abstract

An Investigation into Performance Testing for e-Commerce Web-based Applications

CANDIDATE: Michael J Mhlabane - BCom(Hons)(IS)

SUPERVISORS: Prof D Petkov, Dr T Nepal

DEPARTMENT: Information & Communication Technology

FACULTY: Commerce

DEGREE: M.Tech (Information Technology)

E-commerce Web-based application performance testing is a fairly new concept which makes use of past experience for solving new and challenging problems.

This thesis investigates the applicability of certain software testing methodologies as the basis from which the new field of Web-based application testing can arise. In particular, it concentrates on performance testing issues and Web application testing, as these are responsible for ensuring the survival of a business organization in the new global competitive age presented by the Web and Internet technology.

In this study, firstly, software testing is discussed in general. Particular attention is later paid to software performance testing. The nature and architecture of electronic commerce is also thoroughly studied as a stepping stone towards issues on performance testing. Software performance testing tools are evaluated and one is employed in testing e-commerce Web-based applications. Analysis of the test results obtained is done and conclusions drawn. The premise of this study is that Web site performance should be good enough to be able to cope with the expected number of concurrent users.

Suggestions for a further refinement in Web site performance testing to improve the response time and through-put are not exhaustively investigated in this study as they fall outside the scope of this research.

This research does not aim to substitute any of the existing approaches related to the investigation of software testing techniques affecting Web development, but rather to complement them for the purpose of providing deeper insight into the role of the various factors affecting e-commerce Web-based application performance testing.

Keywords:
Software Testing, Performance Testing, Electronic Commerce, Web Applications, Business-to-business (B2B) e-commerce, Business-to-Consumer (B2C) e-commerce, Automated Testing, Testing Tool.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisors, Professor D Petkov and Dr T Nepal, for the privilege of being able to study under them. Thank you for always being available to give advice and guidance, for encouraging me, for providing your expertise and assisting me with my studies.

A special thank you to Prof. Olga Petkova who, in addition to academic support, also provided lots of advice and guidance.
Another special thanks to the NRF for the financial support, and making my studies a success.

For all the assistance with the statistical and mathematical analysis, I would like to thank my colleagues Corrie Strumpher and 'Ferdo' Maree at Cape Peninsula University of Technology (CPUT). You both helped in putting things in perspective from a statistical/mathematical viewpoint.
I would also like to thank Prof. G. Erwin, at CPUT, for the advice on the technical analysis of the test results. I would also like to specially thank Oliver Manickum for the technical support, advice and the role in this research. Another special thanks goes to Vicky Munsamy and Palesa Seekane for the proof reading of this thesis.

Thanks also to:

- All the participants who contributed to the success of the test experiments of this study.
- The Web site owners for permitting me to use their live Web site as a subject for the test experiments.
- My brothers and sisters, for your support and encouragement through the past years.
- My former colleagues, at Unilever SA and ePages eBusiness Consultants and Developers, for their endless support and to my current colleagues at CPUT for their encouragement.

PREFACE

The research work reported in this thesis was carried out on a part time basis at the Department of Information & Communication Technology, Faculty of Commerce of the Durban Institute of Technology, under the supervision of Professor D Petkov and Dr T Nepal.

This study represents original work by the author and has not otherwise been submitted in any form for any other degree or diploma at any Institution, Technikon or University. Where use has been made of the work of others it has been duly acknowledged in the text. This work reflects the personal investigations and test results of the author, made with Web performance testing software, described later in this thesis. Product names and company names mentioned herein may be trademarks of the respective owners.

Signed: _____

M J Mhlabane

Table of Contents

Chapter 1: Introduction

1.1 Introduction to Software Testing

The advent of the "new platform" of the Internet and the World Wide Web (called 'Web' in short) has presented the software development field with a new dimension of challenges. This new technology has made software development quite different from the traditional software development practices. As Mochal (2001) put it, 'the Web has presented the software development industry with a backwardness in terms of what can be applied or not when compared with the traditional development models.' One of the challenges the Web technology has brought is related to the guidelines to follow when testing software. This activity in software development is considered as an after thought in a number of projects. As a result, the complexity of fixing the errors, if they are ever discovered, is increased. The maintenance of software becomes also more difficult. The need for enhancing Web software testing methodologies was one of the motivations for this research.

Sometimes the software might be well tested for functionality but not tested for other important features. Such features could include performance, usability, navigation, etc, which contribute quite considerably to the success of the product in operation. While presenting such challenges, the Web and Internet medium has certainly fuelled business development and has increased its rate of expansion, and efficiency, enormously. The Internet and its global network of computers have helped establish a global village of networked machines in which communication and information sharing is enhanced (Daily Mail & Guardian, 2001). The Internet provides a variety of services to multiple users anytime and anywhere (Cho, Byun and Sung, 2003).

Effective software development models and processes also require the ability to build into them the fact that testing should not be treated as an after thought, but needs to be considered right from the analysis phase of software development or major software maintenance. Before the software can even begin to be designed and/or developed, it must be known what types of tests need to be conducted and also when and how are the tests going to be conducted. It is therefore necessary to position the issues

regarding software testing and maintenance within the broader field of e-commerce Web application development.

Software testing is well researched with robust theoretical foundations upon which any study of software testing of Web applications should draw. Testing describes the subset of validation and verification activities which involve program code. Validation and verification are two of the most important activities that occur on a software project. The former is concerned with ensuring that an evolving software system matches user requirements. The latter is concerned with ensuring that the output of a project phase is a correct reflection of the input to that phase (McDermid, 1993).

Hetzel (2000) wrote that approximately 50 percent of the elapsed time and over 50 percent of the total cost of a project are expended in testing programs or systems being developed. Testing is an activity most developers have experienced, associated with a great deal of time and money. Despite this fact, most people still find it a bit difficult to define what testing is.

Testing is an unnecessary and unproductive activity if its sole purpose is to validate that the specifications were implemented as written. If the development processes for software work correctly, they would implement those specifications as written. Thus, testing as performed in most organisations is a process designed to compensate for an ineffective software development process (Perry, 1995). During the development process software must be tested for defects to find areas where the program does not conform to its specification and that it meets the users requirements.

There are a number of testing types that can be performed once the development has been completed. Amongst these is performance testing. The open availability of credible measurement and test methods is an important step towards assuring the quality of software-based systems (Wakid, 1999). The following section proceeds with a brief analysis of past work on software testing to allow the formulation of the goal of the research.

1.2 A Brief Introduction to Past Research on Software Testing

The notion of "testing programs" arose almost simultaneously with the first experiences in writing programs. The programs that ran on the first computing machines had to be tested, and references to program testing in the literature can be traced back to 1950 (Hetzel, 2000). Testing was a routine activity associated with engineering and manufacturing processes in industry, and it was quite natural to see it take shape also as part of the software development process.

The software testing process shows much more than how good the application has been coded. It also shows how well the business requirements have been met and how well the application has been designed. The test process has a life cycle of its own.

It is worth stating the obvious about the reason we do testing. First, one wants to ensure that the solution meets the business requirements. For instance, if a specific business transaction follows a logical part, one wants the testing to prove this, in fact, happens. Second, one tests to catch errors and defects. These may be programming errors or errors that were introduced earlier in the developments life cycle. This activity was carried out because there was a need to prove that the solution is correct with no errors or defects.

There are number of tests that one can perform during the testing process. Amongst others, the following are types of testing that can be conducted. These testing types are listed according to classification of software testing according to three criteria.

1.2.1 Classification According to Development/Implementation

- Unit Testing
- Module
- Subsystem Testing
- Integration Testing

- System Testing
- User-acceptance Testing

1.2.2 Classification According to Endurance to Exceptional Conditions

- Workload Testing
- Stress Testing
- Performance Testing

1.2.3 Classification According to Vulnerability

- Online security Testing
- Physical security Testing

When dealing with software testing it is also important that it be checked on how software testing originated. At the same time it is also necessary that e-commerce web site testing be explored in a detailed manner. From the different software testing classification already examined above, this study focused on the "testing according to endurance to exceptions", and more specifically, performance testing. In the case of performance testing, one might need to make sure that the Web application can handle the customer transactions and not leave them hanging for a long time. Basically the throughput must be such that the customer does not have to wait for an unreasonably long time before a transaction is processed. Since this study is about testing of e-commerce Web-based applications, and the focus is on performance testing, the question of what performance testing is needs to be investigated more rigorously in the next chapters.

Electronic commerce is normally written as e-commerce in short. According to Teo and Tan (2002) the phenomenal expansion of computer networks, notably the Internet, has resulted in the rapid proliferation of electronic commerce. E-commerce

is not something new. It has been around for many years. In this technologically advanced age, documents flow automatically in and out of their respective databases; and this mostly does not involve paper, phone calls, or faxes. That was the promise of electronic data interchange (EDI) implemented through Internet technology.

Commercial interests are the largest segment of the Internet and will continue to fuel its growth. The advent of the Web has made it far easier for companies and other organisations to find new ways to expand their business and operations. There are many ways in which companies are exploiting the broadening repertoire of e-commerce technology. Companies are using e-commerce tools to globalise operations, offer personalized customer service, manage sales and support, and even create new business. Meanwhile, the technology continues to evolve rapidly. New applications using the emerging Extensible Markup Language (XML) standard are providing users with added capabilities such as information retrieval on a scale not possible before. E-commerce technology can help erase the traditional boundaries of time and space. E-commerce applications have to be built fast to stay ahead of the changing technology. They must be built well, because breakdowns frequently mean lost business.

Testing is the obvious way to ensure quality applications. But testing tools are difficult to use because most of them require hand-coding of all the scripts (Purpura, 2000). Any company conducting e-commerce on the Web needs to take Web application performance testing seriously.

E-commerce has increased significantly over the last two years in spite of the overall slowing of the economy. This in large is due to improved connectivity. Despite the improved connectivity the Zona Research (2001) report has discovered that online shoppers are still abandoning sites due to poor Web site performance, and that measuring Web site performance is an area often overlooked by Web site owners and developers. The key findings of the Zona Research report (2001) include:

- up to R250 billion potential lost revenue due to Web performance issues;

- business-to-business (B2B) sites have improved average page response times by 50 percent in the past year, yet business-to-consumer (B2C) sites have deteriorated by up to 20 percent;
- average page response time is 17 seconds for many retails sites, as a consequence of focusing on graphics and banner advertisements.

An important issue to remember when doing performance testing is scalability: the ability of the system to handle significant heavier workloads than are currently required (Weyuker and Vokolos, 2000). This necessity might be due to such things as an increase in customer base or an increase in the system's functionality. Either of these two changes would typically cause the system to have to be able to provide significantly increased throughput. If there has not been appropriate testing to assure that the system can be scaled, unacceptable response times might occur as workloads increase.

There has been little research published in the area of software performance testing for e-commerce Web sites. The large amount of available articles focus on testing the Web sites for quality content, without focusing on the performance of applications driving the sites. If testing is at all done, it is only performed on the correctness of the code and that the product meets the business requirements. The performance of the applications is often overlooked. This was another motivation behind this project.

1.3 Statement of the Goal of the Research

Due to the Web the software development methods and the entire software development process changed and new methods/techniques/guidelines/procedures for testing of software are needed. Most of the software used on the Web is either faulty or not usable to its full capacity. Some of the applications are usable, but when it comes to performing at high optimum levels they fail. This is depicted by either crashing Web sites or slow speed with which e-commerce Web site users have to cope with in order to complete their transactions. A number of online traders failed because

of poor e-commerce Web site performance. The latter frustrates the users and they are not getting back to such sites. These issues motivated the goal of this project.

1.3.1 Main Goal

The main goal of this research is to investigate the ways and techniques used to test e-commerce Web-based applications/software, with particular attention on performance testing, and conduct experiments using those techniques, analyse and discuss performance testing results, and draw up conclusions for improvement of Web performance testing.

1.3.2 Sub-Goals

The sub-goals of the research are:

(1) to investigate the foundations and current research issues on software testing;

(2) to investigate the main issues in performance testing and measurement;

(3) to investigate the role of testing on Web-based applications;

(4) to research the issues of performance testing on Web application used on e-commerce Web sites;

(5) to apply suitable software testing methodologies (tools, techniques and guidelines), and principles on experiments to be conducted for this study;

(6) to conduct e-commerce Web application test experiments based on propositions defined for performance testing, finding out if these propositions could be supported by the findings of this study, and document the results from the test experiments for analysis and discussion purposes, and drawing up conclusions thereon;

(7) to investigate the performance rate of a Web site in meeting the recommended performance levels, and the possibility of predicting levels at which interventions could be necessary when performance has degraded beyond acceptable levels.

Information or answers to these issues will help developers, researcher and practitioners in e-commerce Web application development, and further the research for e-commerce Web-based applications.

1.4 Scope, Assumptions and Delimitations of the Research

This research is about the ways and methods used in testing performance of applications used on e-Commerce Web sites on the World Wide Web. The experiments to be conducted on the e-commerce Web sites will be on existing and operational Web sites.

In this study, when software performance is mentioned it will mean to refer to all the activities involved in the evaluation of how the system can be expected to perform in the field. This is considered from a user's perspective, and is typically assessed in terms of throughput, stimulus-response time, or some combination of the two.

It is assumed that the application/software driving the site will continue to be produced in the future in a similar and comparable format. Other applications might complement or support it; or even replace it in the future. It is assumed that the Web site, with its supporting software, will continue to be accessible over the Internet. The reason for choosing to limit the scope to testing of performance on Web-based applications only, and not the other components of the Web system is that most of the Web performance problems reside in the application server. Figure 1.1 below illustrates this point.

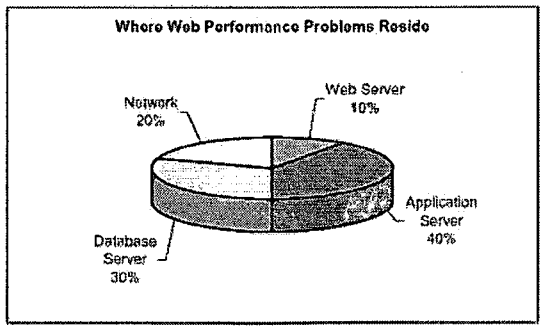

Figure 1.1 – Where Common Web Performance Problems Reside (Empirix, 2003)

1.5 Research Methodology

When undertaking research, it is important to make use of a structured research methodology to ensure that the research has integrity, i.e. that the research is reliable, valid and can be reproduced (Remenyi and Williams, 1993). For results to be accepted they must be reproducible and this is particularly true of laboratory experiments (Squires, 1985). According to Leedy (1989), a research methodology is "an operational framework within which the facts are placed so that their meaning may be seen more clearly."

Remenyi and Williams (1993) suggested three approaches which are appropriate for the scientific acquisition of knowledge in the study of information systems:

- passive observation;
- uncontrolled intervention; and
- deliberate intervention.

These approaches are not going to be discussed further in this study, but are adopted for herein. The nature of the tests which are going to be conducted in this study requires an application of all these three approaches.

A useful triad for justification of research (Robey, 1996) includes research aims, theoretical foundations and research methods. Research aims determine both the theoretical foundations and the research method, whereas theoretical foundations also determine the research methods (see Figure 1.2).

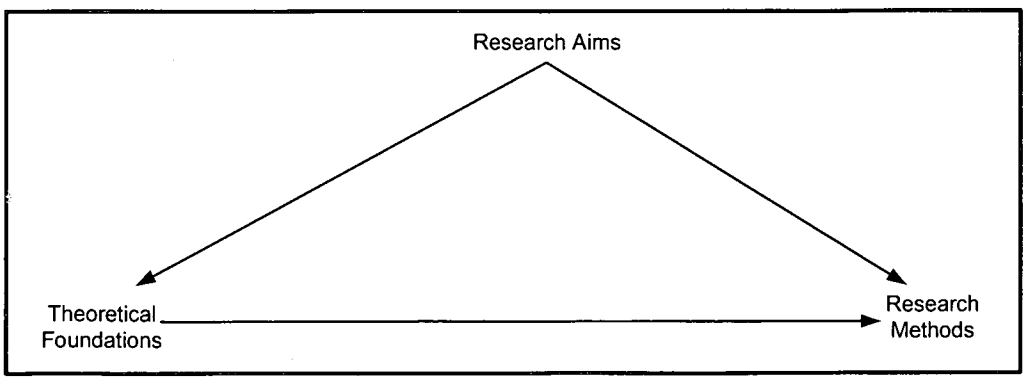

Figure 1.2. A triad on the justification of a research methodology (Robey, 1996)

The starting point in determining the appropriate research approaches are the aims of the research as outlined in the preceding section. The aspects of software testing processes that are considered here provide a deeper insight into the processes of software testing and their foundations. One of the sub-goals of this research is thus to conduct a detailed analysis of the literature sources in the areas of software performance testing and performance measurement. The surveys examine (1) the current state of software testing after 40 years of existence; (2) the foundations and current research issues in Web application testing in general, and more specifically in the e-commerce software performance testing and measurement. The theoretical foundations of this research are formulated on the basis of the literature analysis.

The method for performance testing in Web application development is in the form of a laboratory experiment as that is the most appropriate environment for controlling some of the parameters of the research. In the case of this study, the variables cannot be fully controlled. Nevertheless, the experiments are conducted in the laboratory because of the fact that this is an exploratory study. This experiment represents partial replication of the work of other researchers (Azizi, 2002; Wyuker and Vokolos, 2000), whose results are quite recent within the field of Web development. This research embarks on verifying the conditions laid out in their studies using the Web testing tools, which are not investigated in previously published research in South Africa to the best knowledge of the author.

On the other hand, one of the outcomes from the literature survey is the indication of a need to expand the research methods used beyond the traditional laboratory experiments where appropriate. For this reason the process of gathering remote user feedback in performance of the e-commerce Web environment is investigated through a qualitative field study as it provides more insights into the issue if compared to laboratory experiments. The above considerations outline the research methodology adopted in this dissertation in line with the general guidelines expressed in Figure 1.2

For the purpose of this research, the following activities will be carried out:

- Literature survey on software testing and e-commerce Web sites through the use of books, Internet, journals, proceedings and papers; and
- Creative research and experiments conducted on an e-commerce Web site.

As reported by Remenyi and Williams (1993), it is important for a researcher to evaluate the available literature on the subject being researched to review the current theories and models made by other researchers in the field and to identify unsolved problems. Melville and Goddard (1996) state that a research should be well understood and that a full demarcation of the research problem is also necessary. Therefore the research methodology chosen is based on the goals and scope of the research.

1.6 Significance of the Research

According to Weyuker and Vokolos (2000), there are a number of different goals one could have when designing a research project related to performance testing. There are number of recent publications which deal with the factors which impact on the appropriateness of software performance. On the other hand there are no such publications based on the current South African context. From the available literature, there is no evidence that an attempt has been made to research software performance for e-Commerce Web application in South African organisations. With the absence of any significant research effort in this topic in South Africa, this research begins to fill the gap with a study of the software performance testing for e-Commerce Web applications in South Africa.

1.7 Overview of the Dissertation

This dissertation is structured in the following manner. The first chapter is an introductory chapter, articulating the goals of the research, its importance and justification and broadly its foundations on the theory of software testing. The second chapter explores research issues surrounding software testing in general. The third chapter is a literature survey of Web based e-Commerce, Web application performance testing and its current status. The fourth chapter explores the theoretical foundations of software performance and how it impacts on the Web environment.

Chapter five is focused on test planning and experiment design. Automated software testing tools are also discussed in chapter five. The documentation and testing observations, including the analysis and discussions, are in chapter six. Finally, chapter seven concludes the research and looks at future issues.

Chapter 2: Current Research in Software Testing

2.1 Introduction

This chapter examines the issues of software testing in general. The question of what is software testing, and how it affects system development, is explored. The relations of software testing and other topics and/or disciplines are also examined. Software testing is often equated to finding bugs. However, test scenarios that do not reveal failures are also informative. The purpose of testing a software system is to determine whether it matches its specification and executes in its intended environment (Whittaker, 2000). The fact that the system is being executed distinguishes testing from code reviews, in which source code is read and analysed, usually by a developer or a group of experts. Testing, on the other hand, requires a running executable program or system.

What makes a study in software performance testing for e-Commerce Web based applications important? Ratajczyk (2002) wrote that it is about time that we have testing studies specifically for the Web. Whittakar (2000) also wrote that software companies face serious challenges in testing their products, and these challenges are growing bigger as software grows more complex. Software conformance, assurance, and performance are three key attributes of its quality. For each of these attributes, the metrology-reference, measurement method, and uncertainty statements varies considerably (Wakid, 1999). The first and most important thing to be done is to recognise the complex nature of testing and take it seriously. Software Testing is arguably the least understood part of the development process (Whittakar, 2000). The software testing process shows much more than how good the application has been coded. It also shows how well the business requirements have been met and how well the application has been designed. A test task is to simulate interaction between software and its environment (Whittakar, 2000). The problem comes in where, according to Glass (2001), some practitioners think that all the topics of testing are about its management, and apparently do not know anything about the technical details that can make this topic so challenging.

An often neglected, but extremely important part of the software development process is testing the product to ensure that the developers made no obvious errors (Harrison, 2000). If a Web site is to be used by a large number of users, it is important to ascertain that the site can sustain increased numbers of concurrent users. Many organisations are seeking to deploy mission-critical Web sites and applications that are intended to attract large numbers of visitors, often with significant revenue associated with the activity (Ratajczyk, 2002).

2.2 Examination of the Meaning of "Software Testing"

There have been a number of different definitions of software testing given by the different writers who wrote about software over a number of years. The definitions have also evolved with time and the advancement of technology. For example, Hetzel defined testing as *the process of establishing confidence that a program or system does what it is supposed to (Hetzel, 1973).* Later on, Myers (1979) defined testing as *the process of executing a program or system with the intent of finding errors.*

Hetzel (1988) wrote, in his publication which was an outgrowth of his first formal conference on software testing, that they established, in 1972, the view that "testing" encompassed a wide array of activities all associated with "obtaining confidence that a program or system performed as it was supposed to."

This view of testing makes "finding errors" the goal. Myers (1979) emphasises that if the goal is to demonstrate that a program has no errors, then the 'tester' is subconsciously steered toward that goal; that is, the tester will tend to select test data that have a low probability of causing the program to fail. On the other hand, if the goal is to demonstrate that a program has errors, the test data will have a higher probability of finding errors and there will be more success in testing.

Hetzel's (1988) criticisms of Myers' (1979) definition were that it is too narrow and is also restrictive to be accepted as a definition of testing. His criticisms were based on the fact that there are many ways to test a program or system besides executing it, and

if Myers' definition was taken literally it would mean that testing would only be done *after* a program was coded. Hetzel (1988) adds that the understanding of testing is built on the notion of "measuring" or "evaluating," not trying to find things wrong.

In 1983 Hetzel revised his definition of software testing as *any activity aimed at evaluating an attribute or capability of a program or system and determine that it meets its required results* (Hetzel, 1988).

In software testing, the goal is to help assess software effectiveness (by providing information such as defects found, inspection or test results, etc.), and hence the test and evaluation activities are themselves measurement activities.

Myers' definition again found support when Arthur (1988) provided his version of the definition of testing. *Testing is the process of executing a program with the intent of finding errors. A test that finds an error is a successful test* (Arthur, 1988). The objectives of testing are to:

- Ensure compliance with the original requirements and the approved changes.
- Ensure a quality product. Testing is a major component of a successful quality control plan.
- Reduce the risks inherent in computer systems.

Laudon and Laudon (1998) defined testing as *the exhaustive and thorough process that determines whether the system produces the desired results under known conditions. The Testing determines the validity of the computer solution to a business problem.* According to Louisa (2002) *testing is part of verification and validation. Validation testing aims to demonstrate that the software functions in a manner that can be reasonably expected by the customer.* Verification testing is similar to the above definition Laudon and Laudon (1998).

Wikipedia (2002) defined software testing as *the process used to identify the correctness, completeness and quality of developed computer software.* Testing the system involves development and execution of a test plan in accordance with procedures and user requirements. Actual testing occurs during the construction stage

of development. Testing is the development step in which testers repeatedly hammer on the product to see what breaks it. The developers write code fixes. Then, there is another round of testing to see if the fixes caused anything else to break (Schmeiser, 1999).

2.2.1 A Brief Historical Background of Software Testing

In order to have a better understanding of what testing is, one might have to briefly look at where is testing coming from. The word *test* is derived from the Latin word for an earthen pot or vessel (*testum*). Such a pot was used for assaying metal to determine the presence or measure the weight of various elements, thus the expression "to put to the test" (Hetzel, 1988). Today the word (test) is very commonly employed. Extending the concept from a series of questions or exercises to measure the knowledge and skills of individuals to the measurement of computer systems is where most of us encounter fuzziness (Hetzel, 1988).

The notion of "testing programs" arose almost simultaneously with the first experiences in writing programs. The programs that ran on the first machines had to be tested, and references to program testing in the literature can be traced back to 1950 (Hetzel, 2000). Testing was a routine activity associated with engineering and manufacturing processes, and it was quite natural to see it take shape as part of the software development process.

The early view was that you "wrote" a program and *then* you "tested and debugged it." Testing was considered a *follow on* activity and embraced not only the effort to discover errors but also to correct and remove them. A number of the earliest papers on "testing" actually concerned "debugging," and the difficulty of correcting and removing the errors was long thought to be the more interesting problem. It was not until 1957 that program testing was clearly distinguished from debugging.

During the late 1950s and 1960s software testing came to assume more and more significance because of both experience and economics. It was evident that computer systems contained many deficiencies, and the cost and impact of fixing or recovering from these problems was substantial. More emphasis was placed on "better testing"

by users and development managers, and the time and resources devoted to testing and debugging increased sharply.

In 1972 Hetzel organised the first formal conference devoted to software testing, which was held at the University of North Carolina, and it brought together a broad section of people interested in software testing (Hetzel, 1988).

2.3 Classification of Software Testing

There is a wide variety of software testing types, and within these there is a number of options available to the tester to ensure that the application will be tested thoroughly and meet the standards of today's market. There is a lot of confusion when it comes to the classification of testing types. The classifications may also depend on whether one is testing the software or not. To understand testing and what tests need to be conducted for which system, and at what stage it is important that software testing be categorised into proper classes. The sections below examine these classes.

2.3.1 Classification According to Various Stages of Development /Implementation

With the system development life cycle (SDLC), there are four general phases of system development. According to Satzinger, Jackson and Burd (2004, 2002), the phases of SDLC are (1) Planning, (2) Analysis, (3) Design, and (4) Implementation. Although each application system is different, most errors are found during the integration and system testing (Dennis and Wixom, 2000). Below is an examination of the different testing types conducted within this category.

(a) Unit Testing

Unit testing tests individual software components or a collection of components (Whittaker, 2000). Unit testing is often the first step in testing software. Unit testing focuses on one unit – a program or a program module that performs a specific function that can be tested – and ensures that the module or program performs its

function as defined in the program specification (Dennis and Wixom, 2000). Its intent is to build confidence in the correctness of a unit. At the unit level a tester is mostly interested in algorithmic aspects, verifying whether each unit performs its required function. Unit testing frequently uses test cases selected using the component's actual source code, in which case it is called program-based or white-box testing. Unit testing is generally done by developers who have access to the source code and are familiar with its details, and therefore can constructively use this information (Weyuker, 1998). According to Louisa (2002) unit testing considerations can test interface, local data structures, boundary conditions, independent paths, and error handling paths. Tests conducted at this level are black-box testing and white-box testing.

(b) Black-Box Testing

Black-box testing is also known as *functional testing*. Black-box testing treats programs as a black box. This is normally done at the unit testing level for system implementation. Functional testing is where the tests are derived from the program specification. It requires the selection of test scenarios without regard to source code structure (Whittaker, 2000). According to Kolawa (2000) black-box testing verifies that each class behaves according to specifications. Black-box testing alludes to tests conducted at software interface. Thus, black-box testing examines some fundamental aspects of a system without paying attention to its internal structure(Hamptonu, 2002). Another reason for doing this that black-box testing complements white-box testing (Lousia, 2002).

(c) White-Box Testing

White-box testing is also known as *structural testing*. It looks inside the program to test its major elements. In this case the tests are derived from knowledge of the program's structure and implementation. According to Kolawa (2000) white-box testing tests the construction of one's code. White box testing is predicated on close examination of procedural detail. Logical paths through the software are tested and the internal execution state of the software may be examined at various points to determine if it corresponds to what is expected (Hamptonu, 2002).

(d) Integration Testing

A unit may not be a complete program by itself. The next step in testing process is the integration of units. Why should a program, built from units that work properly when tested individually, not function when tested after two or more units have been integrated? This is primarily because of the differences in the types of errors that one discovers at various levels of testing. The goal of integration testing is to put the units in their intended environment and exercise their interactions as completely as possible. Integration testing assesses whether a set of modules or programs that must work together do so without error. Under this type of testing there is *user interface testing, use scenario testing, data flow testing and system interface testing.*

(e) System Testing

System testing typically uses test cases selected without reference to the code details, because at this level, there is generally far too much code to rely on such details. A test strategy that does not rely on code details is called a *black-box* or *specification-based testing* (Weyuker, 1998). People other than the code developers usually do system testing. They may, therefore, be unfamiliar with the level of detail necessary to perform code-based testing, and generally do not have access to the source code. They are only responsible for testing the fully integrated system.

(f) User-acceptance Testing

The users primarily conduct acceptance testing, with support from the project team. The goal is to confirm that the system is complete, meets the business needs that prompted the system to be developed, meets set quality criteria and is acceptable to the users. Acceptance testing is done in two stages: *alpha testing,* in which users test the system using made-up data, and *beta testing,* in which users begin to use the system with real data but are carefully monitored for errors (Dennis and Wixom, 2000). Beta testing is a process of conducting a thorough, but not completely comprehensive, test pass on software at a major milestone in reference to stability and feature completeness (Tester's Network, 2002).

As part of implementation, *validation* and *subsystem testing* could also be done to ensure the quality of the application or system and confirm conformance to specified requirements and therefore acceptance.

(g) Regression Testing

Regression testing verifies that one's modifications corrected problems and did not introduce new problems into one's code (Kolawa, 2000, Whittaker, 2000). Once a product has been tested and fixes have been made to a product, the product needs to be tested again to ensure that all reported bugs have been fixed. Regression testing is a way to ensure that no new problems have occurred during the fixes. According to Wikipedia (2002), regression testing is a type of software testing where the tester checks that previously fixed faults have not re-emerged.

(h) Interface Testing

Interface testing tests the exchange of data with other system. Since data transfers between systems are often automated and not monitored directly by the users, it is critical to design tests to ensure they are done correctly.

(i) Conformance Testing

In conformance testing, the reference is the standard or specification; the measurement method prescribes a test configuration, a platform type, and test cases. This, therefore, mean that conformance is the process of testing that an implementation conforms to the specifications. According to Wikipedia (2002) conformance testing is testing to determine whether a system meets externally specified standards.

In his list of *testing strategies* Arthur (1988) stated several types of testing types, which are based on their process features. These are:

(i) Gray-box testing

This type of testing combines the strategies/approaches of both black and white box testing.

(ii) Top-down testing

This is the testing type which works incrementally through a program or system, beginning at the top. For the purpose of this study top-down testing will be used, because it works incrementally through a program or system, beginning at the top,

instead of 'bottom-up testing' which begins with the lowest-level modules and works incrementally back up through the hierarchy

(iii) Bottom-up testing

It begins with the lowest-level modules and works incrementally back up through the hierarchy. This is useful when more than one programmer is working on a program change.

The types of testing in this category are briefly discussed in this chapter. They will be discussed in more details in chapter four.

2.3.2 Classification According to Endurance to Exception

From time to time a system might get exposed to conditions which are beyond the system's capabilities. Thus it is important to know the maximum capacity and the capabilities of the system. Testing the systems endurance to exceptions would assist in this regard. The following issues will be discussed in more detail in chapter 4. They are mentioned here only for the need of completeness of the discussion on software testing:

(a) (Work)load Testing

Load testing occurs after the system's functionality has been thoroughly tested. This phase determines whether the system's resource allocation mechanism function correctly and how the system behaves under particular loads.

(b) Stress Testing

Stress testing is intended to test the software with abnormal situations. It attempts to find the limits at which the system will fail through abnormal quantity or frequency of inputs (Louisa, 2002). According to Systest (2001) stress testing finds out how much data the application can process.

(c) Performance Testing

This type of testing, which is the focus of this study, examines the ability to perform under high loads. High volumes of transactions are generated and given to the system. This test is often done by using special-purpose testing software, and this is exactly what is going to be carried out during the test experiments of this study.

2.4 Links Between Software Testing and other Related Topics/Disciplines

2.4.1 Software Testing and Software Quality Assurance

Software is becoming more and more a strategic factor for a lot of industries to be competitive in the market. The capability to develop high quality software within a tight schedule increasingly determines the success of a software organisation (Houdek and Kempter, 1997). To do business successfully an organisation has to improve its software process so that product quality could be improved in a systematic and continuous manner.

The subject of software quality assurance and that of software testing are, to an extent, misused because one has applications which never get well tested, and when the question of "tested" application arises the answer is that the application has been tested, while in actual fact only the application quality assurance department only passed the application for conformity to quality standards. According to the IEEE Software (1998), the subject of software quality is not well defined. Satisfying technical requirements, conforming to standards, and producing the product efficiently are the things which are often not distinguished from each other.

Most recently the topics 'testing' and 'quality assurance' have been topics handled in one conference, and are to an extent merged. One example of this is the recent Practical Software Quality Techniques (PSQT) conference which was held in Bloomington. It was held in conjunction with the Practical Software Testing Techniques (PSTT) (Glass, 2001). Testing has already been examined in detail above. Quality Assurance (QA) will be defined and then a close look taken on how and why

the two (testing and QA) are taking a new meaning and the tendency for them to be discussed together.

Why take a close look at testing and quality assurance? Software quality is a serious issue for developers and customers *(Onoma and Yamaura, 1995)*. With the advent of the Web content managers have turned too much of their attention into quality assurance. Quality Assurance (QA) is the term they use for the process of checking and inspecting their Webs for bugs, errors, flaws, etc.

Is quality assurance another synonym for testing? Is it possible that the use of QA for inspecting Webs for errors is the result of the differences in the Web development and tradition software development?

Glass (2001) wrote that unlike the traditional software development, Web application development tends to be:

- **Requirements-free.** There are seldom any stated requirements at the outset of a project. (One speaker described that phenomenon, only partly in jest, as "Specs? We don't need no stinkin' specs!")
- **Rapid-development.** There is seldom time to do anything more than what one speaker called "slam it and jam it," using a process he called "FAD—Frantic Application Development."
- **Short fuse, relatively small.** Most projects are expected to last about three months, using a maximum of four people.
- **Testing-primitive.** There are few generally accepted processes, and the tools that *are* available often don't fit the problems well.
- **User-intimate.** Although you never meet your users, they are only a mouse click away from letting you know what they don't like—and from going over to the competition.

- **A "living project"** (two speakers used this term). Maintenance, usually by the developer, is a "must," and in fact, "all products are in permanent beta test" (modifications are ongoing).
- **Performed with limited management support.** Most managers think Web development is easy.
- **Performance is vital.** And extremely unpredictable
- **"Content is king."** It doesn't matter whether the application is "pretty" or not if the content is not interesting and/or useful.
- **A whole new field.** Developers are more "Wild Web developer" than software professional.

Is it pure backwardness to build Web applications without requirements, or is it simply true that in any rapidly evolving discipline, it is impossible to know what the requirements are until the first (several?) prototypes are built? And in the absence of firm requirements, is it not impossible to perform the rudiments of traditional testing and quality assurance, such as requirements-driven testing? These questions will be explored further in chapter 3: *Evolution of the World Wide Web.* In the attempt to answer the question on QA, the focus will be on defining quality in software. Dromey (1996) defines quality using traditional quality factors such as correctness, reliability, usability, and maintainability. While, on the other hand, Lauesen and Younessi (1998, p 70) define quality factors through *quality defects* observed during product use. That is, when the product fails, it does so as a result of a quality defect, which indicates a weakness in one or more of these factors.

Something interesting to note is that a defect is not a defect until such time that it surfaces as a result of the software/application use. Some quality defects may not have been observed because they have not caused failures. Using test cases that cover all the functional and non-functional specifications it is possible that the quality defect can be trapped using the test process before the application has been put in the hands of the users. According to Onoma and Yamaura (1995, p68) this statement can be

supported because improving quality assurance can significantly decrease the fault rates in software products.

2.4.2 Software/System Testing and Software Life Cycle

There are many case studies showing that the software testers should be involved in the software design and development phases, yet the industry continues to treat software testing as an afterthought, and testers as unskilled temporary positions that can be filled by persons with no special training (Ideva, 2002). Fraternali and Paolini (2000) also wrote that a careful review of the Web site features revealed that most solutions concentrate on implementation, paying little attention to the overall process of designing a Web application. This section serves to show that there are strong links between the software development life cycle and software testing, which should then manifest the importance of software testing. According to Whittaker (2000) software testing is arguably the least understood part of the development process.

There are two types of software testing. One involves the information services (IS) department and confirms that the change complies with their specifications. Testing software systems built using a life cycle development methodology will adequately address this type of testing. The second involves users, where the intent of change is tested. Software testing though the SDLC is discussed below.

(a) Testing and Requirements Analysis

The major developmental activities that take place during this phase are the elicitation and clarification of requirements and the subsequent construction of the system specification. The major testing activity that occurs during this phase is the derivation of the verification requirements. These are requirements which, during the latter stages of the project, are converted into the system and acceptance tests: tests which determine whether a system meets user requirements.

Although the verification requirements can be established as late as the final stages of system design, it is important that they be established as early as possible during the requirements analysis stage. It is during this stage that the test plan should be developed.

(b) Testing and System Design

There are a number of activities that are carried out during system design which are relevant to testing. First, the verification requirements will be expanded out so that they correspond more closely to individual test.

The second reason is that it provides a check that the system design has been correctly developed. By tracing verification requirements against program units, staff charged with quality assurance are rapidly able to discover inconsistencies and omissions in the system design.

The fourth reason is that it checks whether the verification requirements which have been processed and expanded during system design have been expanded to the right level of detail.

A final reason is that it provides useful information during the process of system modifications. When the program code, in a specific unit or units, is changed the developer wants to know what other functions may be affected by that change. The usual reason for this is that he/she wishes to run a series of tests to ensure that a modification due to a requirements change has not affected other functions of the system.

(c) Testing and Detailed Design

The next stage of the software project is detailed design. The main testing activity that occurs during this phase is the construction of the test procedures. A test procedure is a details step-by-step set of instructions for the staff who carry out the final stages testing.

(d) Testing and Programming

The first, or primary, activity in this phase is programming or coding the individual units or modules. Work may also be carried out on producing test harnesses or stubs.

A second activity is the testing of the program units after they have been programmed. This process, normally called unit testing, is an informal process. The word 'informal' does not imply disorganised of unstructured. It means that the testing

process is not under the control of a quality assurance professional; it is normally carried out by the programmer who produced the unit.

(e) Testing and Integration

Testing during the integration phase will follow the plans set out in system, or architectural, design. The primary aim of the testing activity is to verify the design, but a subsidiary is to begin to verify requirements functions.

2.4.3 Software Testing and Software Maintenance

Software maintenance is a task that every development group has to face after the exhilarating rush of getting the product out the door. What makes it particularly difficult is that maintenance requires applying a different lifecycle model and includes its own unique challenges. Software maintenance is routinely the poorly managed headache that nobody wants to deal with (Carr, 2000). Software maintenance is viewed as a problem because of the amount of budget it is taking from software budgets. Instead of being devoted to making improvements software maintenance is devoted to fixing errors. Lifecycle data shows that maintenance is where the biggest chunk of practitioner time and money is spent (Glass, 1998). If the software was well tested and bugs fixed in the first place during the development stages no money would be wasted in software maintenance.

Maintenance costs over a software system's lifetime are generally two to three times higher than the development costs (Osterweil and Clarke, 1992). Maintenance covers a spectrum of activities, including bug fixing, streamlining, changes and enhancements. All these require altering the basic software, and so all necessitate retesting and reanalysis to assure that the altered software satisfies changed requirements. In the wake of this problem, it can be difficult to deal with this problem because according to Tan and Gable (1997, 2000) previous IS research has shown that problems related to maintenance staffing are among the more important problems in software maintenance. In particular, the attitudes of maintenance personnel toward their work can have adverse effects on their performance and the quality of the application portfolios they help to sustain.

Life cycle data shows that maintenance is where we spend the biggest chunk of practitioner time and money (Glass, 1998). Testing is as important to software maintenance as it is to new system development. Frequently, even small changes require extensive testing. Given that modifying old software to serve new purposes requires testing, it is not unusual to spend more time testing a change and training users to operate a new facility than incorporating the change into the application system (Perry, 1995). Once one has converted the software, traditional issues about its architecture will arise. Performance, reliability, and security will be the three of the biggest problems (Horowitz, 1998). Ideva (2002) attested to this when they wrote "our software systems have become so complex that no one can accurately predict all the ramifications of making a change to one."

Too frequently software maintenance has been synonymous with "quick and dirty" programming, which is rarely worth the risk. Frequently, it takes considerable time to correct problems that could have been prevented by adequate testing.

According to Arthur (1988) testing in a maintenance environment is similar to testing in a development environment. The principal differences are:

- Only changes need to be reviewed, not the entire product.
- Only new test cases that exercise the changes need to be developed. They can then be added to the existing test bed.
- Only the existing test cases associated with changed products, along with any new test cases, will be needed to test the changes.
- Test results can be mechanically compared against previous test results to identify variations in the output caused by the changes. This comparison process is commonly known as regression testing.

Testing takes up to fifty percent of the development budget and a similar amount of the maintenance budget to ensure a high-quality product (Arthur, 1988). Therefore, it is essential to plan the testing process adequately during system development so that test maintenance can be reduced to minimum levels.

It is easier to test the system incrementally, as designs, code, and programs are revised. This is called *incremental testing* and is based on the principle of incremental testing where testing integrates human and computer testing to test the documentation and software in small chunks. One advantage of incremental testing is that maintainers can begin testing early in the maintenance cycle and eliminate many defects in the requirements and design phases. By integrating the system in stages, a narrow set of interfaces is tested one at a time. This "build a little, test a little" philosophy is a way to maximise productivity and minimise risks associated with maintenance. Incremental testing spreads the work over the maintenance process, rather than concentrate it at the end. This prevents reliability problems in the delivered product. Each maintainer can unit test his/her own modules while everyone else codes and tests theirs. A test plan should identify the strategy for testing each component at each phase of the maintenance process (Arthur, 1988).

2.4.4 Software Testing and Performance Requirements

Mochal (2001) wrote that in the absence of firm requirements it is impossible to perform testing and ensure quality in a requirements-driven testing. Requirements are the foundation of the software release process. They provide the basis for estimating costs and schedules, as well as developing design and testing specification (Stark *et al*, 2000). Weyuker and Vokolos (2000) wrote that in order to do software performance testing in a meaningful way, it is necessary to have performance requirements. This should be explicitly included in a requirements or specification document and might be provided in terms of throughput or stimulus-response time and might also include system availability requirements. Frequently no such requirements are provided, which means that there is no precise way of determining whether or not the performance is acceptable.

When requirements have been agreed on by both clients and management, then adding to, deleting from, or modifying those existing requirements during the execution of the software maintenance and testing processes impacts the maintenance costs, schedule, and quality of the resulting product. In practise there can be a problem because some software developers tend to ignore requirements that do not agree with their creative development sensibilities. Thus it is not unusual for a marvellous software product that the developer created to be a poor fit for the

software buying market. While it is true that a developer's creativity may be inhibited by his or her knowledge of the facts, the developer's creation can be crippled by ignorance of the facts.

Understanding and identifying requirements and their impact on software development is imperative for the successful delivery of software products. This can translate into software testing in that one has to know what is required of the software in order to successfully test it. Performance requirement represents the effort that goes into managing the performance requirements for information systems. Since information systems have to provide an interface with external users, both functional and non-functional requirements have to be considered. Unlike other types of systems (e.g., operating system software), e-Commerce web sites (and other information systems) have to maintain a persistent database, handle incoming data and transactions, and provide an interface with external users. To be effective, such systems should meet both functional and non-functional requirements. *Functional requirements* describe what a system should do, typically stating the intended functionality in terms of input-output relationships. By comparison, *non-functional requirements (NFRs or software quality attributes)* deal with such software quality factors as performance, accuracy, security, reliability, and robustness (Nixon, 2000). Non-functional requirements consider software quality factors as security and reliability (Cheng *et al*, 2000).

This study proposes the treatment of performance requirements as non-functional requirements, and applies performance issues to test the performance of e-commerce Web based applications. Cheng *et al* (2000) point out that performance testing is fundamentally and qualitatively different than merely testing for correct function, walking through the real-life issues associated with systematically testing for performance. These include setting objectives, the need to test for scalability, choosing measurements to gather, what it means to measure resource utilisation and contention, how to generate workloads, the role of requirements and specifications, and many others. It is for this perspective that this study will conclude with a comprehensive performance test experimentation, with enough detail to show the issues (and some possible ways for their resolution) in actual practice.

Performance requirements need to be managed, and this can be a difficult task due to the nature of performance requirements. First, performance requirements can have a global impact on the target system (Nixon, 2000). Meeting one requirement may change several parts of a system. One cannot simply add a "performance module" to produce a system with good performance. Rather, one must consider performance requirements throughout the system and throughout the development process. With requirements satisfaction the focus is on achieving the functional and performance objectives of software system. Poor quality in this sense means errors and discrepancies with the requirements.

The above discussion on performance requirements examined mainly instances where there are problems related to lack of performance requirements. In concluding this section it is necessary to note that Weyuker and Vokolos (2000) wrote that another crucial issue when doing performance testing is making sure that if there are stated performance-related requirements, they can actually be checked to establish whether or not they have been fulfilled.

2.4.5 Software Testing and Project Management

According to the Project Management Book of Knowledge Guide (2000) project management is defined as the application of knowledge, skills, tools, and techniques to project activities to meet project requirements. A key step in project management is work planning, which is the process of identifying the tasks required to accomplish objectives, assigning responsibilities and estimating completion dates (Larson, 1995). Many tools support the management of various development activities, including configuration management, security, testing, and team development. A few tools support very complex projects with developers in multiple locations, so that the Internet itself becomes a vital part of the development environment (Chmura and Sharon, 1996).

Weyuker and Vokolos (2000) wrote of their findings on projects which had performance related problems. They discovered that ninety three (93) percent of performance-related project-affecting issues were concentrated in a thirty (30) percent of system that were deemed to be problematic. They concluded that projects that are in good shape are very likely to recognise that addressing performance issues are a

necessary part of the architecture phase of development, while those projects that are in the most trouble almost never have what they often perceive to be the "luxury" of considering performance issues. Performance problems account for many of the show-stopping faults encountered once the projects are released to the field and, so, developing comprehensive testing strategies is essential.

Although project management is an old concept, the new medium of Internet and the Web has presented a new paradigm in which project managers are forced to revisit their project management techniques and redefine them to suit the new challenges presented by the Web and Internet technologies. Flexible delivery of products based on automated change control is a key activity in project management, which is a new challenge for traditional project management practices. Discussions held by researchers and some project management professionals has highlighted problems that are typical of many software-testing organisations. This includes the need for more effective software testing tools, lack of communication between individuals working on similar projects, low status of some project management personnel, and lack of design-for-project management philosophy during the software development phase. According to Mochal (2001) the testing process has a life cycle of its own, and shows how well one met the business requirements and how well the application is designed. Project management is all about planning and scheduling activities with time and cost estimates. If considered, according to Mochal (2001) as already stated above, a project manager would consider testing and plan and schedule time and cost for testing, and this would ensure better success of the project.

2.4.6 Software Testing and Software Reliability

Some systems might not be used frequently enough to warrant extensive testing. However, critical systems need to be tested for reliability, regardless of their infrequent use. Most reliability engineers recognise the need to cater to uncommon "critical functions," such as those used infrequently but having extremely severe consequences if they fail (Kitchenham and Linkman, 1998). Such systems should be tested for operation functionality and reliability. This is beneficial in that it provides one with a measure of software's reliability; and is often stated in terms of the probability of failure in a given time period.

According to Ntafos (1998) the main method that is used in order to reach some level of confidence on the reliability of software is program testing. Program strategies (testing criteria) execute the program on a (usually very small) subset of its inputs. Herein lies the problem and challenge of software testing. Thus, the question of how can one select the test cases so that one can be confident about the reliability of the program from a very small sample? But then again one would still question the reliability on software whose test was based on a small sample.

2.5 When to Consider Testing

Some commercially developed software systems go through little or no systematic testing, which can lead to serious consequences once the software has been released to the field (Weyuker, 1998). The later in the life-cycle that software faults are identified, the greater the cost of repair and more serious the impact on the end user. Most testing, analysis, and debugging work is ad hoc and done manually. Humans are relied on to use their intuition and judgment to create test cases, evaluate results, and decide when testing is sufficient. Nontrivial automated support is the exception, not the rule. There are no widely accepted formal guidelines or standards for determining when testing and analysis is adequate (Osterweil and Clarke, 1992). As a result, the development teams in different organisations would test at different stages/phases of the development cycle. Some work is needed in this field.

2.6 Practitioners' View of Software Testing

Glass (1998) wrote that there is a wrong understanding of software testing. This means that there is a need to study software testing in depth and give it the necessary attention it deserves. Hetzel (1988) gave the following list, which basically illustrated how software testers in practice view testing. These views give a proper way to look at testing as a broad and continuous activity throughout the development process.

- *Checking programs against specifications*
- *Finding bugs in programs*

- *Determining user acceptability*
- *Insuring that a system is ready for use*
- *Gaining confidence that it works*
- *Showing that a program performs correctly*
- *Demonstrating that errors are not present*
- *Understanding the limits of performance*
- *Learning what a system is not able to do*
- *Evaluating the capabilities of a system*
- *Verifying documentation*
- *Convincing oneself that the job is finished*

2.7 The Testing Process

During the testing process, careful control and management of test information is critical. Except for small programs, systems should not be tested as a single, monolithic unit. Large systems are built out of sub-systems which are built out of modules which are composed of procedures and functions. The testing should therefore proceed in stages where testing is carried out incrementally in conjunction with system implementation. According to Sommerville (1997) the most widely used testing process consists of five stages, as shown in the figure below. In general, the sequence of testing activities is component testing, integration testing then user testing.

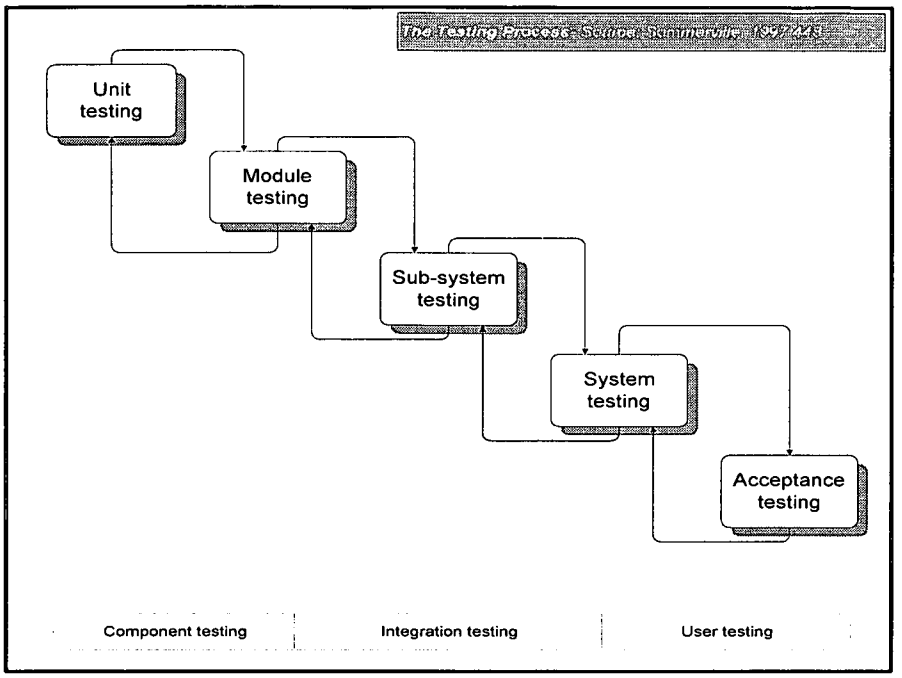

Figure 2.1 The testing Process (Sommerville, 1997)

Sommerville (1997) also presented what he called generic model for defect testing process. The figure below depicts this process.

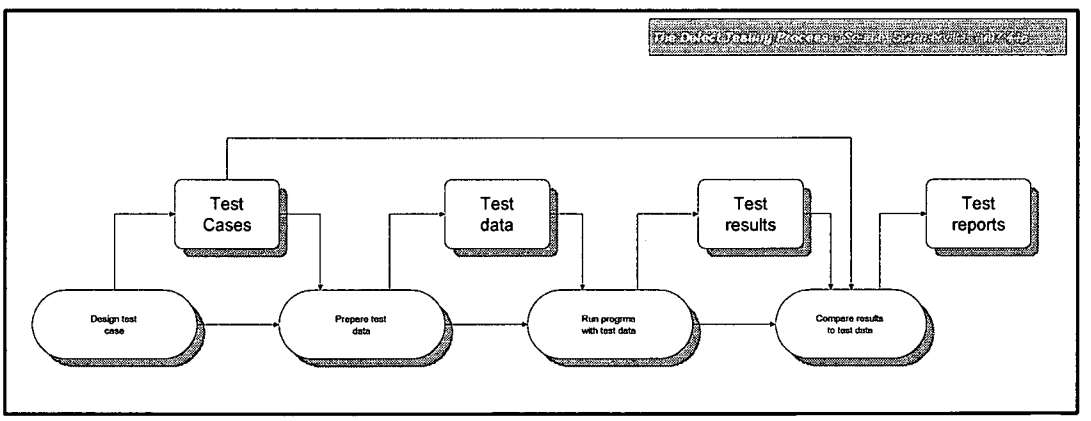

Figure 2.2 The testing Process (Sommerville, 1997)

Defect testing is intended to exercise a system so that latent defects are exposed before the system is delivered. This contrast with validation testing which is intended to demonstrate that a system meets its specification. A successful defect test is a test which causes the system to perform incorrectly and hence exposes a defect. This

emphasises an important fact about testing. It demonstrates the presence, not the absence, of program faults.

Defect testing is not intended to show that a program meets its specifications. If the test suite for a program does not detect defects, this means that the tests chosen have not exercised the system so that defects are revealed. It does not mean that program defects do not exist.

2.8 An Examination of General Approaches for Evaluating Software Testing

To help deal with software testing; there is a need to offer a structured and systematic approach. It would be desirable to use principles that address the issues of consideration in software testing, rather than an ad hoc approach.

To deal with the body of knowledge which includes software performance concepts and systems development techniques, there is a need for catalogues of knowledge to collect and organise it and make it available to the people who can benefit from it, like, for example, developers, consultants, managers, software engineers, etc.

In dealing with the problems of test evaluation, researchers are pursuing two approaches, which are (a) formalism, and (b) embedded test code (Whittakar, 2000). In explaining the two approaches Whittakar (2002) wrote that 'formalism' chiefly involves the hard work of formalising the way specifications are written that designs and code are derived from them. Both object oriented and structured development approaches contain mechanisms for formally expressing specifications to simplify the task of comparing expected and actual behaviour. Industry has typically shied away from formal methods; nonetheless, a good specification, even an informal one, is still extremely helpful.

There are essentially two types of embedded test code. The simplest type is test code that exposes certain internal data objects. As implemented, such functionality is

invisible to users. Tester can access test code results through, for example, a test debugger.

A more complex type of embedded code features self-testing programs. Sometimes this involves coding multiple solutions to the problem and having one solution check the other.

2.9 Chapter Conclusion

This chapter examined the issues on software testing in general. A number of different definitions of software testing by different writers were examined. Performance testing was examined. Links and relations of software testing with other topics and disciplines were also established and it has been discovered that software testing is related and linked to a number of other topics and disciplines. Software testing has also been classified under three main categories.

The next chapter will examine issues related to Web and Internet technologies. The effects of performance on e-commerce Web-based applications will also be examined.

Chapter 3: Evolution of the Web/Internet and Performance Issues

3.1 Introduction

Advancements in Internet and Web technology merit attention in today's Internet dependent world. Advanced communication technologies are bringing enormous changes to business organisations by offering easier ways for customers to purchase goods, as well as to their service providers to sell goods to a wider audience than before. In the early days of the Web, just being able to navigate through the pages of a Web site, jump from one site to another with a mouse click, was a major breakthrough. This interaction model that was based on the single click of the mouse, not even the double click, was responsible for the tremendous success of the Web. After a few years this model was proven inadequate, because it did not allow viewers to interact with the Web server in the way they interacted with typical Windows applications (Petroutsos, 2000).

The Web was based on the premise that clients requested documents from the Web server. This was a reasonable assumption for the early days of the Web, but the unexpected adoption of this technology led very quickly to the need for a more elaborate scheme of information flow between clients and servers. Users are no longer interested in simply requesting documents from a server. Users need up-to-date, live and instant information. They need to search databases, look up goods' prices, place orders, and send, as well as retrieve, information from the server.

This chapter focuses mainly on the purchasing of products and/or services over the Internet using electronic commerce Web sites. From this point onwards, electronic commerce will be referred to as e-commerce. The next sections will examine e-commerce further by looking at its background, in depth definitions, commercial use, etc.

3.2 Background Issues on e-Commerce and Web/Internet Technology

The advent of personal computers in 1983 and the spread of LANs and client/server architecture in 1989 put the infrastructure in place for the explosion of the World Wide Web over the Internet in the early 1990s. Together, the various technological developments and forces have coalesced to bring the Internet, through the Web, to millions of people. Along with millions of pages of HTML that provide information and entertainment, the idea of e-commerce over the Internet has begun to hold as a viable way of doing business (Chapman, 1998).

To many people, the concept of e-commerce is very new. Mentioning e-commerce to many people is like talking Internet or the Web. One would ask if e-commerce is only Internet or Web, and without the two technologies would it exist? According to Chapman (1998) the history of e-commerce began in the 1970s with the introduction of electronic funds transfer (EFT). Electronic data interchange (EDI) and e-mail were widely introduced in the late 1970s and early 1980s. The 1990s have seen the commercialisation of the Web on the Internet. This technology brought the arrival of wide-spread e-commerce. E-commerce can occur over the Internet, on intranets, or across extranets. E-commerce can be business-to-business (B2B), business-to-customer (B2C) or government-to-business/consumer (G2B/G2C). The B2B and the B2C concepts will be defined later in this chapter.

The Internet's origins trace back in the early 1960s, when the US Department of Defence became very concerned about the possible effects of nuclear attacks on its computing facilities. The US Defence Department realised that the weapons of the future would require powerful computers for coordination and control. The powerful computers of that time were all large mainframe computers, so the Defence Department began examining ways to connect these computers to each other and also to weapons installations that were distributed all over the world. The Defence Department agency charged with this task hired many of the best communications technology researchers, and for many years funded research at leading universities and institutes to explore the task of creating a worldwide network that could remain

operational even if parts of the network were destroyed by enemy military action or sabotage. These researchers worked hard to devise ways to build networks that could operate independently; that is, networks that would not require a central computer to control network operations.

In 1969, researchers connected computers at the University of California at Los Angeles, Stanford Research Institute International, the University of California at Santa Barbara, and the University of Utah. Over subsequent years, many researchers in the academic community connected to this network and contributed to the technological development that increased the speed and efficiency with which the network operated. At the same time, researchers at other universities were creating their own networks using similar technologies.

The Web will become even more important as there is a shift toward e-commerce, and product software becomes Web-based software (Powers, 2000). This has the effect of making software testing more complex and even more important than ever. In less than thirty years the Internet has become one of the most amazing technological and social accomplishments of the century (Schneider and Perry, 2000). Millions of people are using a complex, interconnected network of computers. These computers run thousand of different software packages. These computers are located in almost every country of the world. Every year billions of monies change hands over the Internet in exchange for all kinds of products and services.

The opening of the Internet to business activity helped increase the Internet's growth dramatically. However, there was another development that worked hand in hand with the commercialisation of the Internet to spur its growth. That development was the World Wide Web, called Web in short. It was introduced in the early 1990s.

The Web is more a way of thinking about and organising information storage and retrieval than it is a technology. As such, its history goes back many years. The two important innovations which played a key role in making the Internet easier to use and more accessible to people were hypertext and graphical user interfaces. In the past, code was developed for hypertext server programs and was made available on the Internet. Hypertext servers used on the Web today are usually called Web Servers,

and present information in the form of HTML documents. Web browsers were developed to read HTML documents. A Web browser presents an HTML document in an easy to read format in its graphical user interface. The number of Web sites has also grown even more rapidly than the Internet itself, and is currently estimated to be well over eight million. The number of Web Documents is likely to be over a billion (Schneider and Perry, 2000). Each Web site might include hundreds or even thousands of individual Web pages.

3.2.1 The Structure of Web Applications

Since this research is about investigating the performance of e-commerce Web-based applications, it is important to look into what exactly a Web application is. As already stated above, the Web started as a global network that simplifies the sharing of information. Some people post information in the form of Web pages, or HTML pages, or HTML documents, and many more people view this information. With the introduction of HTML forms and specialised software that runs on the server, the Web has become an environment for running applications. The Web is no longer a simple click-and-view environment. It has become a client/server environment for running elaborate applications where the browser is the client, which can display HTML documents and interact with the viewer through HTML controls and scripts that runs on the server and the same on the client. The Web server processes the information and sends the results to the client in the form of HTML pages. Nowadays, people talk about Web applications. These applications are known to some people as ASP-based applications, because ASP plays such an important role in developing scripts on the server, though other techniques have been introduced for that purpose.

Nguyen (2000) pointed out that Web applications are complex systems that involve numerous components: servers, browsers, third-party software and hardware, protocols, connectivity, and much more. According to Glass (2001) a Web application is characterised with a typical architecture which includes a browser tied to a Web server tied to an application server tied to a database.

A Web application is a site with multiple HTML pages and server-side scripts. Web applications have a special requirement. Where typical desktop applications use Forms to interact with the user, Web applications are based on HTML pages. A Web

application is a site that works much like a Windows application, and specifically like a client/server application. Viewers enter information on a Form. This information is then transmitted to the server, and the result of the processing returns to the client as another HTML document. In most cases, the processing that takes place on the server is a database search. The results of the search are then furnished back to the client in the form of another page. The major difference between a desktop application and a Web application is that the windows of a desktop application can remain open on the desktop and users can switch from one to the other with a mouse-click. This is not true with a Web application. A Web application can only display one Form (page) at a time. In order to switch to another page one must either select a link on the current page, or click the Back button to view a page that has already been visited.

Lately, business organisations deploy Web applications onto the Web and engage in e-commerce activities. Conducting business online through the use of the Web is essential to organisations hoping to succeed in e-commerce. However, most companies find it difficult to provide customers with prompt and effective online service. This could be due to unexpected system downtimes which blow revenue out of most organisations. How can an organisation insulate a system to guarantee maximum availability and avoid lost business and productivity?

A Web application is characterised by multiple environments, multiple platforms, a new approach, and the focus is different in that one must 'think user'. The goal of e-commerce has been to develop a virtual electronic market that could provide online shoppers and sellers opportunities for online trading. This study focuses especially on the electronic market where the seller sells goods and services to users no matter where they are around the globe. Such electronic markets are Web and Internet technology enabled, accessible anywhere where a user can access a Web site.

The Web and Internet architecture is based on the typical Internet client/server infrastructure and the other enabling applications and hardware. It comprises components such as Web server, and a Web browser-based user interface.

According to Gartner (2001) there is a critical need for uptime in today's global economy, and to survive a business should cope with the twenty-four hours and seven

days a week online demanded availability. This is caused by the increased dependences on computers and the Internet. This also translates to the need for business organisations to ensure that their e-commerce Web-based applications are properly tested, and can withstand the high demand and expectations of the Web community. The next section examines the meaning of e-commerce.

3.3 Defining e-Commerce

The Web has made online shopping possible for many businesses and individuals. E-commerce means shopping on the part of the Internet called the Web. Although consumer shopping on the Web is expected to increase as the years go by, e-commerce is much broader and encompasses many more business activities than just Web shopping. In this broader sense, some people and businesses use the term electronic business (e-business in short) when referring to e-commerce. However, most people use the terms e-commerce and electronic business interchangeably (Scheider and Perry, 2000). Hence, in this study, the term e-commerce will be used in its broadest definition.

At a higher level, there are two major types of e-Commerce applications, namely, business-to-consumer (B2C) and business-to-business (B2B). Initial interest in e-commerce was in the B2C sector. There are numerous examples of high-profile B2C Web-commerce applications, including Amazon.com, CDnow, and 1-800 Flowers, Kalahari.net, and countless other mail-order businesses. These are internationally popular B2C e-commerce Web sites.

Most experts agree that the real impact of Web-based commerce will be in the business-to-business sector. For this study, the focus is on products that provide solutions for business-to-consumer commerce over the Web.

B2B e-Commerce is nothing new in principle. EDI has been around for years, but it is expensive and beyond the scope of all but the largest companies (Patel *et al*, 2002). With the Web, e-Commerce is now viable for companies of any size, not just

corporate powerhouses with deep pockets and vast resources. The Web opens new opportunities for companies to engage in commerce with their suppliers, customers, government, and business partners. However, there are different classes of e-Commerce applications within the online business sector. The primary classes are sell-side, which focuses on selling goods or services to consumers; buy-side, which focuses on helping companies make procurements; and marketplace, in which an aggregator brings together multiple buyers and sellers, providing a community for commerce among the participants (Patel *et al*, 2002). Before examining deeply what e-commerce is all about, traditional transactions and online transactions will be briefly examined for their differences.

3.3.1 Traditional Commercial Transactions

The origins of traditional commerce occurred before recorded history when our ancestors first decided to specialise their everyday activities. Commerce or doing business, is a negotiated exchange of valuable objects or services between at least two parties and includes all activities that each of the parties undertakes to complete the transaction. The parties to any transaction normally involve the buyer and the seller. In traditional commerce settings the buyer begins by identifying a need. Once buyers have identified the needs they must find products or services that will satisfy those needs. They use a variety of search techniques to find products. These may include consulting catalogues, asking friends, reading advertisements, consulting salespersons, or examining directories. Buyers in traditional commerce contact vendors in a variety of ways, which includes telephone, mail, and attending trade shows.

The seller in traditional commerce often undertakes market research to identify potential customers' needs. Advertisements are placed and buyers purchase goods over the counter. The seller has to engage in business processes which includes cashing all money and making arrangements to deposit the revenues at the bank.

3.3.2 Online Commercial Transactions

With online transactions the picture is different from that of traditional transactions. The buyer goes online using the Internet to search and find goods or services. The seller has to have a Web site from which online shoppers can buy products or services. In this case, the buyer cannot feel or touch the product. Products online are viewed by means of pictures/images and mere descriptions. A buyer will view the product and choose to buy it. One needs to realise that there are multiple intrinsic elements that interact with each other during an online (e-commerce) transaction. Included in this mix is the customer computer, the browser, plug-ins, scripts (CGI, JavaScript and even Java applets), the Internet, the Web, firewalls, Web servers, databases, legacy systems, bank and credit card networks, and even telephone lines (Ocampo, 1999). Together there is ample opportunity for problems to occur that are possible risk and threat to data integrity. This also opens up a question on the performance of the different components that have to work together at a speed that does not negatively impact on the customer's online shopping. The risks and the performance issues involved in such transactions will be examined later in this chapter.

3.3.3 Electronic Commerce

In the past years that people conducted business with one another, they have adopted the tools and technologies that became available. More recent innovations, such as the printing press, the telephone, and other innovations, each changed the way in which people conducted commerce activities.

Firms have used various electronic communications tools for decades to conduct different kinds of business transactions. All kinds of businesses have used EDI to place orders and send invoices, and retailers have used television advertisements to generate telephone orders from the general public for all kinds of merchandise.

A good definition of electronic commerce would mention the use of electronic data transmission to implement or enhance any business process. Some people use Internet commerce to mean electronic commerce that specifically uses the Internet or the Web as its data transmission medium. Since the field of e-commerce is so new to some businesses, people and businesses sometimes use terms in different ways. For

example, IBM has defined electronic business to be "the transmission of key business processes through the use of Internet technologies" (Schneider and Perry, 2000).

While Patel *et al* (2002) wrote that at a higher level, there are two major types of e-commerce, namely business-to-business and business-to-customer; Shelly *et al* (2000) wrote that e-commerce businesses could be grouped into four basic models, as follows:

- Business-to-Consumer, referred to as B2C
- Business-to-Business, referred to as B2B
- Business-to-Employee, referred to as B2E
- Market-to-Market, referred to as M2M

(a) Business-to-Consumer/ B2C

B2C consists of the sale of products or services from a business to the general public or end user. Many B2C companies surveyed by Teo and Tan (2002) usually have their established businesses before venturing online. Hence, they may still be experimenting with their Web business and will reasonably exercise caution when making investments in their cyber-stores. According to Patel *et al* (2002) the primary purpose of B2C applications is revenue generation, where an organisation sells products or services to multiple buyers.

(b) Business-to-Business/B2B

B2B is not new. EDI has been around for years – but it is expensive and is beyond the scope of all but the largest companies (Patel *et al*, 2002). With the Web, e-commerce is now viable for companies of any size, not just corporate powerhouses with deep pockets and vast resources. In B2B the Web opens new opportunities for companies to engage in commerce with their supplier and business partners. The primary focus of B2B is on the buy-side, in which companies make procurements. Of the vendors supplying tools for B2B commerce, Ariba Technologies Inc. provides a rich feature set for these kinds of services (Patel *et al*, 2002).

(c) Business-to-Employee/B2E

B2E mainly runs on the business organisation's Intranet or Extranet. Such applications allow companies to run internal sales and purchases by company personnel.

(d) Market-to-Market/M2M

Market place solutions are a relative newcomer to e-commerce, and very few online marketplaces exist today (Patel *et al*, 2002). Marketplace applications create a virtual community that brings together multiple suppliers and multiple buyers. In this case, the market place provider supplies the application infrastructure that let multiple parties transact business.

According to Patel *et al* (2002), e-commerce applications have four distinct layers: low-level system services (load balancing, scalability, fail-over, database connection pooling, etc.), specific business logic (catalog functionality, inventory-handling capabilities, or shipment request and tracking), administration (which can also include modules for target marketing or personalisation), and payment processing. No matter what application one is trying to implement, the application must include these layers. Systest (2002) wrote that order entry, customer support, bill payment and similar systems were traditionally client-server based. They may have had to contend with as many as 500 trained, somewhat fault-tolerant users simultaneously demanding system time. With today's business becoming increasingly available on the Internet, these same systems now serve up to millions of less computer-sophisticated users, all vying for time to place orders, pay bills and more. These users expect one's system to work correctly and almost instantaneously, or they will go to the competitor's Web site.

This study does not focus primarily on the use of electronic data transmission technologies, but on testing the performance of the Web sites driven by electronic data transmission for the purpose of online trading.

3.3.4 Role of e-Commerce

One opportunity that many businesses are finding as they examine their industry value chains is that e-commerce can play a role in reducing costs, improving product quality, reaching new customers or suppliers, and creating new ways of selling existing products. Web based e-commerce also afford businesses an opportunity to have their virtual doors open for twenty four hours and seven days a week.

By examining elements of the value chain outside the individual business unit, managers can identify many business opportunities, including opportunities that can be exploited using electronic communications technologies and e-commerce.

When firms are considering e-commerce, the value chain can be an excellent way to organise their examination of business processes within their business units and in other parts of their product's life cycle. Using the value chain reinforces the idea that e-commerce should be a business solution, not a technology implemented for its own sake.

e-Commerce agents: e-commerce based agents are still in their infancy. Some of the well-known ones help users find information about products or services. The user provides information directly or indirectly. Based on this information, the agent searches for a matching product or service (Lee and Lee, 1997). These agents are changing the manner in which business is conducted in the new information age.

3.3.5 Positive Elements Associated with the Use of Web-base e-Commerce

There are some elements that have a positive influence on e-commerce. According to Schneider and Perry (2000) organisations are interested in e-commerce because it can help increase profits. These writers summarised the e-commerce for business entities as: 'e-commerce can increase sales and decrease costs.' Turban *et al* (2000) wrote that e-commerce benefits organisations, consumers, and the general society by expanding the marketplace to national and international markets. The costs of handling sales inquiries, providing price quotes, and determining product availability can be reduced by using e-commerce in the sales support and order-taking process of a business. Whelan (2002) wrote that three factors compelling investments into e-commerce are

(a) a high profitable business model; (b) rising barriers to entry; and (c) huge market opportunity.

Just as e-commerce increases sales opportunities for the seller, it increases purchasing opportunities for the buyer. Businesses can use e-commerce in their purchasing process to identify new suppliers and business partners. Negotiating price and delivery terms is easier in e-commerce, because the Web can provide competitive bid information very effectively. E-commerce increases the speed and accuracy with which businesses can exchange information, which reduces costs on both sides of transactions.

E-commerce provides buyers with a wider range of choices than traditional commerce, because they can consider many different products and services from a wider variety of sellers. This wide variety is available for evaluation 24 hours a day, every day. Some buyers prefer a great deal of information before deciding on a purchase, while others prefer less. E-commerce provides buyers with an easy way to customise the level of detail in the information they obtain about a prospective purchase. E-commerce can make products and services available in remote areas. Despite having such notable positive elements, e-commerce also suffers some drawbacks. These are examined below.

3.3.6 Negative Elements Associated with Web-based e-Commerce

Schneider and Perry (2000) discussed the disadvantages associated with e-commerce. These writers stated that some business processes may never lend themselves to e-commerce. For example, perishable foods and high-cost items such as jewellery or antiques may be impossible to adequately inspect from a remote location, regardless of the technologies that are developed in the future. Most of the disadvantages of e-commerce today, however, stem from the newness and rapidly developing pace of the underlying technologies. These disadvantages will disappear as e-commerce matures and becomes more available, and accepted by the general population. Many products and services require that a critical mass of potential buyers be equipped and willing to buy via the Internet. This is evident in that some of the e-commerce services available online as offered by some businesses have delivery services being made available to

selected few cities. This will change as more customers become connected to the Internet and begin to feel comfortable with purchasing online.

Businesses often calculate return on investment before committing to any new technology. This has been traditionally difficult to do for investments in e-commerce technology, because the costs and benefits have been hard to quantify.

Many businesses have trouble recruiting and retaining employees with the technological, design, and business process skills needed to create an effective Web-based e-commerce presence. Another problem facing businesses that want to do business on the Internet is the difficulty off integrating existing databases and transaction processing software designed for traditional commerce into the software that enable Web-based e-commerce.

In addition to technology and software issues, many businesses face cultural and legal impediments towards e-commerce. Some consumers are still somewhat fearful of sending their credit card numbers, and related information, over the Internet. Other consumers are simply resistant to change and are uncomfortable viewing merchandise on a computer screen rather than in person. The legal environment in which e-commerce is conducted is full of unclear and conflicting laws. In many cases, government regulators have not kept up with technologies. Laws that govern commerce were written when signed documents were a reasonable expectation in any business transaction. As more businesses and individuals find the benefits of Web-based e-commerce to be compelling, many of these technology and culture related disadvantages will disappear.

3.4 Issues Related to the Web-Based e-Commerce Downfall

The relaxation and easing of restrictions on Internet commercial activity saw a start in implementation of plans to privatise the Internet. Network Access Providers started selling Internet access rights directly to larger and indirectly to smaller firms through companies known as Internet Service Providers (ISPs). The Internet was a phenomenon that truly sneaked up on an unsuspecting world (Schneider and Perry, 2000). People outside the researching community were largely unaware of the potential offered by a large interconnected set of computer networks.

Some of the Internet technologies were designed at a time when researchers thought Internet traffic would be predictable and stable. However, with the popularity of Internet protocol applications, analysts predict that public network traffic will increase by thirty times over today's traffic in the next few years.

According to Chapman (1998) e-commerce is very attractive to both business and consumers. Chapman's article was written in the year 1998, time by which there was still all the good talk about e-commerce. The fall of the 'dotcom' caused a lot of concerns and scepticism to a number of organisations, even to those who were still planning to launch their business online. Since the time when this article (Chapman, 1998) was written, e-commerce has seen many ups and downs. Some maintain that the e-commerce revolution is just as profound as the changes that came with the industrial revolution (Turban *et al*, 2000). The next sections examine the growth trends, resistance to the usage of e-commerce Web-based applications, security concerns, and the future of e-commerce in the new economy.

3.4.1 The Resistance to Using Web-based e-Commerce and its Decline

Since its inception, traditional commerce has dug its roots deep into the goods and services provision. Consumers came to accept the culture of doing business the traditional way. Moving an established culture from a well accepted way of doing business to a Web based method of doing the same thing takes time, determination, and a good cost benefit ratio (Ideva, 2002). For some consumers, and business organisations, trading on the Web amounts to cultural change, and introducing culture

change is a risky business. The "dotcom" fall caused many businesses and individuals to be sceptical of investing and trading over the Internet. Many of the reasons for failure were a result of many people with motivations fuelled by the search for the "quick buck," rather than the search for lasting customer value (Dalgleish, 2000).

According to Palmer (2002) Web site success concentrated heavily on the frequency of use, user satisfaction, and intent to return as key measures of the user's success with the Web site. The failure of some Web sites to perform according to users' expectations led to some online shoppers abandoning many of the online shopping markets. The online sellers attempted to capture the online shoppers' style and taste, by invading the buyer's privacy. According to Filos and Ouzounis (2002), currently, the most accepted technique is the 'cookie' that is being stored into user machines without prior knowledge of the user. This violates the privacy right, buying habits and interests of users. At the same time fails to provide enriched personalised services.

Tomorrow's e-commerce winners will be the sites that help customers accomplish their goals better, faster, and more easily (Dalgleish, 2000). The next section examines the issue of security, which is closely related to the reason some Internet users are reluctant to buy over the Internet.

3.4.2 Security Issues as the Major Cause of Resistance to Buying Online

One of the most common problems that keep customers away from shopping online, is the infiltration of computer systems. There are many aspects of Internet security and e-commerce security. Most are beyond the scope of this research, but some of the elements of security include Web server security that uses correctly implemented firewalls, correctly implemented proxy servers and correctly implemented internal security for corporate networks. Security concerns is one of the major reason many customers resist the use of Web-based B2C services. Security can be a big issue (Powers, 2000). According to Chapman (1998) security, and the assurance of it, is now the most important issue on the Internet in terms of the development and wide-spread use of e-commerce on the open network. Security determines the ability of application to resist unauthorised entry or modification (Ocampo, 1999). According to

Cho *et al* (2003) quite a few studies employed benefit and risk factors in their analysis of shoppers' willingness to purchase over the Internet.

Security issues must be addressed for all the key areas of the application, such as the browser, network system, application pipeline, Web server and application server, the operating system itself, the database, and other partnered systems connected to the whole application. Sceptical and potential online shoppers get even more worried when they read statements like the one from Chapman (1998), where he wrote that *"there is no definitive way to keep data hundred percent secure when it is sent over the Internet. The Internet is not a secure environment and there is no way to completely guarantee privacy in cyberspace."* From time to time the media fills with news of hackers having broken into some databases and accessed information on credit card holders. According to Cho *et al* (2003) hacking has been around since the early days of phone systems or mainframe computers, and with the development of network technologies, hackers now break into any computer. The greatest challenge of e-commerce over the Internet is the development of confidence in the ability of all involved to offer a better guarantee of data integrity or security. The role played by the media on news reporting makes this challenge even harder. Elliot and Fowell (2000) insisted that for the continuing growth of e-shopping, there should be an improvement in customers' expectations on transaction security.

3.4.3 Web and Internet Based e-Commerce Research

Lately there has been a sudden burst on the number of e-commerce related research projects. Cho *et al* (2003) mentioned that a number of studies have been conducted on the extent to which the Internet influences us, and e-commerce is probably one of the most popular subjects among many studies on the effects of the Internet. These studies are being conducted because business organisations as well as individuals have recognised that Internet affects socio-economic activities and cultural aspects of both organisations and individuals.

3.4.4 The Growth and Future of Web-based e-Commerce

The positive elements associated with e-commerce have already been examined. This section examines the potential for future growth of e-commerce. The aspect of daily life that has been most affected by the Internet is that of commerce. Despite the negative elements already noted above, e-commerce has been on a steady growth. The growth can be attributed to many sources such as convenience, saving time, cheap prices, vendor information availability, easy comparison, and no implicit pressure to buy (Cho *et al*, 2003). According to Graja and McManis (2001) e-commerce is an increasingly significant part of the global economy. A study conducted by Zona Research (May 2001) indicates that Web-based e-commerce has increased significantly over the last two years, due in large part to improved connectivity. During the last few years Web sites have shifted from displaying electronic catalogues to providing a channel for complete solutions to business challenges, generating sales using interactive and dynamic tools as well as electronic customer service. The growth of Web-based e-commerce consulting has changed the way business is done today. The Internet has become an integral aspect of business in the information age. Web-based e-commerce has grown significantly, and will continue to do so in the years ahead (eMarketer, Inc, 2002).

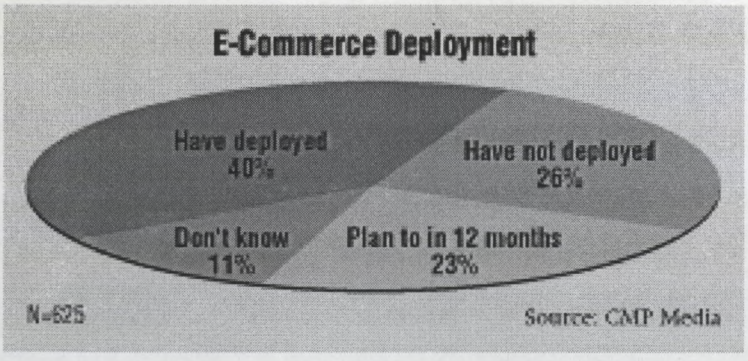

Figure 3.1 – E-commerce deployment by companies (Gartner, 2001)

Figure 3.1 illustrates that 40 percent of companies surveyed have deployed e-commerce. With 23 percent of companies planning to deploy e-commerce in about twelve months this shows that there is potential growth for e-commerce deployment by more companies. It is also possible that those companies falling in the "have not deployed" and the "don't know" sections might decide to deploy e-commerce in the near future. Since this was a 2001 report, it is very much possible that today the "have

not deployed" and "don't know" have already moved to the other parts of the pie-chart.

The above diagram depicts a graph which illustrates that a large percentage of companies have already embarked on deploying e-commerce in their organisations. E-commerce is gaining momentum and is destined for growth in the new electronic economy. The diagram illustrates that whether or not one agree with such estimates of e-commerce growth, there is no doubt that Web-based e-commerce has arrived and is here to stay. The Web provides a whole new sales channel for companies that have traditionally relied on their sales personnel for direct and indirect sales. Other organisations are springing up for which the Web is the only sales channel. Some organisations are even assigning sales quotas to their CIO's, making technology groups responsible for using information systems to generate revenue. Web-based e-commerce gives organisations global reach, and it is far less expensive than alternatives such as electronic data interchange. Web-based e-commerce can also streamline processing, facilitate branding, and lead to greater customer satisfaction (Gartner, 2001).

As more companies jump on the Web-commerce bandwagon, e-commerce products and systems are justifiably being viewed not as technology infrastructure, but as a central part of an organisation's sales and marketing operations. According to Graja and McManis (2001) it is widely accepted that e-commerce activity will continue to grow, and that it will be a significant component of the global economy in the near future. Patel *et al* (2002) wrote that e-commerce is gaining momentum like a runaway truck heading down a mountain pass, and it gives companies global reach, and it is far less expensive than alternatives such as electronic data interchange. Web-based commerce can also streamline processing, facilitate branding, and lead to greater customer satisfaction.

Cho *et al* (2003) wrote that media services and advertising provided through the Internet might affect behaviour patterns related to the traditional media. The hold that the traditional media have exerted on people is being diluted. That is, TV is kind of showy and rich in visual information delivery, radio comes with simplicity and portability, and newspaper conveys in-depth coverage. However, in cyber space there

is media convergence. The quality of visual context over the Internet has been greatly enhanced recently owing to the implementation of high-speed network technology. The online shops are always open. Systest (2001) wrote that in this fast-paced global economy, Web sites have the potential to attract audiences far larger than traditional marketing vehicles. Azavar (2001) wrote that the growth of e-commerce consulting has changed the way business is done today; businesses will spend about R400 billion a year on Internet strategic services by 2005.

A mutual interest exists for both online shoppers and providers. There is no doubt that, owing to all the above strengths, positive elements, optimistic statements by the different writers, and research work conducted on this topic, Web-based e-commerce services will continue to grow in the future.

Despite the rapid growth of electronic commerce and its rapid embracing by more and more organisations, it is important to note that in many cases business processes use traditional commerce activities very effectively, and these processes cannot be improved upon through technology. Products that buyers prefer to touch, smell, or examine closely are difficult to sell using electronic commerce, and this will continue to be a challenge for businesses conducting e-commerce services. In addition the fact that the dot.com bubble burst with certain negative consequences to the IT industry may be seen also as a challenge for rethinking the current practices in e-commerce which may be related to a greater need for attention being paid to issues of testing of e-commerce sites.

The following section will explore issues of testing of e-commerce sites as an activity necessary to improve e-commerce penetration.

3.5 Web Site Testing and Web Application Performance

3.5.1 General Issues on Web site Testing

Unlike mainframe systems applications, Web-based e-commerce operates on a distributed system based on a client-server environment where data and processes are dispersed and replicated across miscellaneous platforms. This networked, cross platform nature requires that one has much tighter process control during application testing and that one pays a lot more attention to traditional forms of software testing such as configuration testing and compatibility testing (Mosley, 1995). The current popularity of Web applications brings with it new challenges because of the wide variety of browsers, technologies and environments. Web technology is complicated in that quality assurance and testing groups must be flexible in order to address the complexity of the applications under test (Ocampo, 1999). Although general quality assurance principles apply in testing Web-enabled applications, some areas of consideration must be examined in order to develop a clear understanding of the necessary testing effort. These areas are application architecture, testing technique, etc.

(a) Web-based Testing Techniques

A test strategy should be devised for testing the Web application. Some of the techniques which could be used are functionality testing, compatibility testing, performance testing, and security testing. These issues will be examined further in chapter four.

(b) The Need to Test Web-based Applications

The new medium of Web and Internet technology has brought about a new type of a user who is not loyal to any Web site. A customer on one's Web site is only a click away gone (Powers, 2000). Graja and McManis (2001) also wrote that users of e-commerce Web sites often have high expectations for the quality of service, and if those expectations are not met, the next site is only a click away. Such statements make it clear that Web site testing has become a critical function of the software testing and quality assurance departments, and therefore needs to be given the attention it deserves.

3.5.2 Testing the Functionality and Usability of a Web site

Any application developed for use brings functionality issues as a cause for concern. In order to test the performance of a Web site one needs to ensure that the site being tested is functional. A good Web site is more than just something to look at, it is functional and interactive. Powers (2000) wrote that the functionality of a Web site must be tested just as the functionality of packaged software.

Web site usability issues are also a cause for concern when one has to conduct Web site tests. If one's site is not user friendly and intuitive, users will quickly go somewhere else, probably to one of one's competitors (Powers, 2000). According to ApTest (2002) usability testing is testing the ease with which users can learn and use a product. To reduce the number of deserters, one needs to start thorough usability early in the development cycle when changes in design are easier to accommodate. The issue of functionality and usability testing will also be revisited in chapter five.

3.6 Web Site Performance Issues and the Effects of Poorly Performing e-Commerce Web Applications

Most systems are tested for functionality and usability, but not for performance (Grossman *et al*, 1996). Grossman *et al* continued to write that performance testing reduces the risk of poor performance at the time of application delivery. According to Mochal (2001) Web performance is vital and extremely unpredictable. Businesses' reliance on software demands that a set of rigorous quality standards be developed and that a much higher degree of professionalism be imposed (Larson, 1995). In a Web environment, users drive functions and exchange content using a variety of standard protocol (HTTP, HTTPS, SOAP, XML-RPC, SMTP, POP3) from any location, using any Internet (browser, Internet enabled application, embedded devices) at any time (PushToTest, 2003).

Visitors to a Web site expect fast results and do not want to wait for delaying downloads. While testing, one needs to be aware of performance and know what can affect performance (Powers, 2000). Internet conditions vary from moment to moment

and this affects performance, but certain issues are directly related to the design of the site. Powers (2000) noted the following as issues of Web site design which are related to the design of the site.

- **Graphics:** These should be small enough and must not cause unnecessary prolonged download times.
- **Load time:** Measure how long it takes for pages to load. Controls, like the 'submit' button, should be tested to see how long a new page downloads and confirm the submit.
- **Connection methods:** Testing should be done using different connection methods. Office and home environment connections should be simulated.

These issues need to be taken more seriously. A study by Zona Research (May 2001) discovered that online shoppers are still abandoning sites due to poor Web site performance, and that measuring Web site performance is an area often overlooked by Web site owners. Cho *et al* (1999) wrote that any business conducting online trading needs to ascertain that its Web site has the sophistication necessary to attract and keep online shoppers and at the same time balance that with high quality performance. According to Glass (2001) the acceptable maximum response time for a Web site is about eight seconds – anything more than that requires telling the user what is going on. Nielsen (2002) stated that the minimum response time is three seconds for a quick loading page, and the reasonably tolerated maximum response time is ten seconds. These acceptable levels of response time will form the basis and used as the yardstick for measuring the performance of the Web site during the experimental tests.

According to Gartner (2001) availability, is the portion of time a system is used for productive work. The level of availability needed varies between Web sites and end users. Online customers services may need 24-hour system access while none-online customers may only require access during business hours. The online category of customers are prime candidates for high system availability solutions, which keep a system free of unexpected or unexplained outages and help manage downtimes. Web site downtime, either planned or unexpected, is unacceptable to online customers, and hence is harmful to an organisation.

Planned downtime usually occurs as a result of system maintenance, repairs, or upgrades. A Gartner (2001) survey listed the following as the reasons for often occurrence of impromptu downtimes.

Table 3.1: Reasons for System Crashes (Gartner, 2001)

Business Failures
CPU failure
Double-bit memory errors
System overload

The performance, speed and reliability of servers and networks is vital to corporate Web sites, but the most reliable network in the world would not help a company whose customers cannot navigate the site and come away thinking the company is doing a poor job of managing it (Knowledgestorm, 2002). Performance testing reduces and eliminates most of the problems associated with poorly performing Web sites, and also point out areas of risk. One needs to remember that as one of the company's major point of contact, a site has the ability to directly affect profits (Systest, 2001). Cho *et al* (2003) concluded that waiting time on the Internet has a negative influence on the consumers' evaluation of Web sites; while Dalgleish (2000) wrote that tomorrow's e-commerce winners will be the sites that help customers accomplish their goals better, faster, and more easily.

3.7 Chapter Conclusion

This chapter has examined the concept of e-commerce/e-business, how e-commerce has helped business organisations improve their profits, and also looked at Web based applications and how these are expected to perform. The effects of poorly performing Web sites were also highlighted as well as those related to the recent decline in interest towards e-commerce research which by no means indicates that e-commerce will have a diminishing role in the future as the above analysis shows. More attention was paid to issues related to testing of web based e-commerce systems, which are further discussed in the next chapter.

Chapter 4: Software Performance Testing and Measurement Issues

4.1 Introduction

The previous chapters examined the nature of the Web and Internet and how business organisations exploit these technologies commercially to conduct their businesses. One of the noticeable items from the previous discussions is that a Web application needs to operate and serve the users with acceptable performance; otherwise business organisations lose their customers. The previous discussion concluded by noting the reasons for abandoning Web sites. One of the main noticeable reasons was that online shoppers abandon some Web sites because they cannot stand the time they have to spend waiting for a site which loads up at a 'snail pace.' As the reason for the slow load-up of the Web sites was identified as poor or bad Web site performance, this chapter focuses on the issues surrounding software performance.

In this study, software performance will refer to all the activities involved in the evaluation of how the system can be expected to perform in the operational environment. This is considered from a user's perspective and is typically assessed in terms of throughput, stimulus-response time, or some combination of the two.

Weyuker and Vokolos (2000) wrote that there has been very little research published in the area of software performance testing. Although there have been very few publications describing approaches to software performance testing, it is nonetheless an extremely significant issue for many large industrial projects. Often, the primary problems that projects report after failed release are not system crashes or incorrect system responses, but rather system performance degradation or problems handling required system throughput (Weyuker and Vokolos, 2000). When queried, it is not uncommon to learn that although the software system has gone extensive functionality testing, it was never really tested to assess it expected performance.

The performance issues which this chapter will focus on are those that are experienced by users for some reasons which are not evident when the application was first

developed and deployed in its operational environment. These do not have to refer to functionality, accessibility and usability degradation. According to Keynote Systems (2002) Internet performance problems are generally not server problems. This basically means that there could be other reasons for a Web site on the Internet to perform poorly. Nevertheless, one of the objectives of this study is to experiment on the e-commerce Web based applications and report on the observations made during the tests.

For the clarity of this study it would be appropriate to define several basic notions in the area of performance testing, surveying some representative sources in this field. This will be done in the following sections.

The Synchronous Optical Network (Sonet) – the standard for wide-area networking – was designed at a time when researchers thought Internet traffic would be predictable and stable (Mehrotra, 2000). Although Internet bandwidth and Web server capacity have improved in recent years, Web site performance problems continue to challenge developers and testers. The combination of complex Web-based applications and the dynamic characteristics of Internet traffic can cause significant degradation in Web site performance.

Performance problems can occur at many points along the route between one's Web site and its visitors (Splaine and Jaskiel, 2001). Some of these performance problems can be predictable and some cannot. The performance problems, which one can predict, can be dealt with in time before online revenue is lost. Performance problems caused by unpredictable events are more difficult to identify and resolve. A Zona Research (2000) report estimated that, in 1999 lost revenue due to unacceptable load time at all commerce Web sites exceeded R362 million per month. There are many consequences of under-performing Web sites. Slow (loading) pages can render marketing campaigns less effective, and "dropped" transactions can result in lost revenue and a number of associated results.

4.2 The Nature and Background Issues of Software Performance Testing

Performance issues account for one of the major fault categories. Performance problems might include such things as the lack of performance estimates, the failure to have proposed plans for data collection, or the lack of a performance plan (Weyuker and Vokolos, 2000). The testing plan is discussed in chapter five. According to Wikipedia (2002), in software engineering, performance testing is testing that is performed to determine how fast some aspects of a system perform under a particular workload. Such testing can be performed purely to demonstrate that the system meets performance criteria, as a comparison between systems to determine which performs better under such circumstances, or, when it is know that a system's performance is less than desired, to determine what parts of the system or workload cause the system to perform slowly. In such diagnostic cases it is common for tools to be used to instrument the program to determine what parts of it take most time to run. In performance testing it is often crucial and equally often difficult to arrange for the test to be performed under similar conditions.

After having published the Web site to the Web it must be tested to discover if it can handle the expected workload. This can be done in a number of ways, like for example, increasing the workload on the systems and have the system stressed to its limits. Systest (2001) defined performance testing as testing to ensure that the system meets all performance requirements and to identify the limits of the system. The conditions under which a system should be tested for performance are normal or nominal conditions, high load conditions, high volume conditions and stress conditions. It is testing conducted to evaluate the compliance of a system or components with performance requirements (ApTest, 2003).

According to Dittmeier (2000) one second can cost an organisation thousand of rands. In confirmation of this statement, a Zona Research (2002) study found that eight seconds is the threshold of an average user's patience when waiting for a Web page to download or a transaction to complete. This means that if an organisation's Web site is slower, it is probable that customers are getting lost. Thus an organisation should

know if its Web site is performing up to par. An organisation should also determine its options if it needs to determine its site performance.

Whether the problem resides on the application page, network segment or server, or some other components of the system it is important to run a performance test to drill down to the root cause of performance problems so that issues can be resolved quickly before end users experience them. This enables operations groups to deliver peak Web site performance and protect the e-commerce revenue stream. Many businesses use performance testing to ensure their Web sites are performing well.

4.2.1 The Importance of Web Performance Testing

From the small information Web site to the large multi-million dollar e-commerce site, Web performance is key to user satisfaction (Gross, 2001). If Web sites only contained static HTML, performance would not be so much of an issue. Web server performance for static contents is largely bandwidth limited, and the technology to scale by adding more Web servers is well known (WebPerfomance, 2002). The picture changes greatly, however, with the addition of scripts and database on the back-end. It is quite easy to write or buy a back end script that is either too slow or does not scale, in which case adding more Web servers simply masks the underlying bottleneck.

A few years ago companies would just wait for someone to call in and report problems with their Web site. Nowadays, the focus is on making sure those problems are detected and fixed before a customer notices (Dittmeier, 2000). Customers do not need to call to let a business know that there is a problem with its Web site. In many cases, the competing business' Web site is probably online, and customers can just go one click away and do business elsewhere. Web performance is directly related to customer satisfaction, and thus monitoring a site is critical. Without performance monitoring the only way to know if a site is down is if users call to complain. People who experience problems such as bad links or poor response times are more likely to leave sites than they are to contact companies with complaints (Gross, 2001). From the user's perspective, the ability to abandon sites when problems are detected is the beauty of the Internet. From the company perspective, losing customers that quickly can be difficult to overcome.

Communicating with customers through the Web and e-mail is essential to business organisations hoping to succeed in e-commerce. However, many companies find it difficult to provide customers with prompt and effective online service. E-commerce has increased significantly over the last four years, which is a result of increased connectivity and technology advancements. Performance experienced by users varies greatly depending on the metropolitan areas in which they are located (Keynote, 2002). The need for performance testing underlines that online shoppers are still abandoning sites due to poor Web site performance, and that measuring Website performance is an area often overlooked by Web site owners (Zona Research, 2001).

The Zona Research (2001) investigation, conducted in the US, had the following key findings:

- Up to R250 billion in potential lost revenue due to Web performance issues
- Business-to-business sites have improved average page response times by 50 percent in the past year, yet business-to-consumer sites have deteriorated by up to 20 percent.
- An average page response time is 17 seconds for many retail sites, as a consequence of focusing on graphics and banner advertisements.

Often customers never come back to a site because it has been unavailable for a number of times or hours. This basically highlights the fact that Web site performance is something which must be given more attention than it currently enjoys.

When developing distributed applications on the Internet, performance testing can often be left until the last minutes, or even put off until it becomes a problem. When one is in the last stages of finishing a Web application or adding features to an existing site, there is no time to delay deadlines to configure and learn how to use a performance testing tool.

4.2.2 Discussion on Whether or not Software Performance Testing is a Wasted Effort

Sometimes it happens that an application, which was developed without being tested for performance, is launched on the Web only to find that the application functions without any errors, and performance is not degraded even during heavy use. Under such circumstances, would it be appropriate to then say that should the application have been tested that would have been a wasted effort?

The Web has been struck by a number of incidences where online shoppers would be unable to access certain Web sites because those sites could not cope with the load (or increased number) of concurrent users. Considering such incidences, the effort should be towards software performance testing standards, which can make a significant contribution towards improving software development and maintenance from art form of software engineering discipline. Putting stress and load testing off to the last minute is common, but it leaves one little time to do anything substantive when one discovers that the product does not scale up to more than twelve users (Marick, 2002).

4.2.3 Application Performance and Recovery Testing

Software is only one component of a system. When developed, the software eventually gets incorporated with other system components and system integration and validation tests performed. On e-commerce Web sites one can carry out recovery testing, security testing, stress testing, load testing and performance testing. Many Web sites need to be fault tolerant, and processing faults must not cause overall system failure. A Web based system requires recovery after a failure within a specified time (Louisa, 2002). Recovery failure is the forced failure of the software in a variety of ways to verify that recovery is properly performed.

As already stated above, users abandon a site due to poor performance. The poor performance could result from a number of reasons, one of which could be an unexpected increased number of concurrent users. When the system reaches its maximum load and crash it is important that it must be recovered in a specified period of time before users get frustrated and abandon the site for other sites. If performance testing was carried out, and the maximum number of concurrent users identified, recovery testing can be carried out to alleviate situations whereby the site can crash

and fail to recover urgently. The incredible task is on running proper tests that can reveal the maximum numbers of users.

4.3 Categorisation and Comparisons of Performance Testing According to Testing Strategy

One of the classifications of software testing discussed in chapter two was classification according to endurance to exceptions. That section is revisited here so that an in depth analysis of performance testing can be carried out. A testing strategy is a general approach to the testing process rather than a method of devising particular system or component test (Sommerville, 1997). Different testing strategies may be adopted depending on the type of system under testing consideration and the development process used. Performance problems are based on events that fall into two broad categories: predictable and unpredictable. Thus, performance tests should be designed around both kinds of events (Splaine and Jaskiel, 2001).

Despite the above facts one still needs to ask: 'what is performance testing?' According to Dennis and Wixon (2000) performance testing examines the ability to perform under high load.' According to Splaine and Jaskiel (2001) performance testing in its broadest sense allows one to observe and evaluate a Web site's response under all possible load conditions for all possible periods of time. Aptest (2002) defined performance testing as "testing conducted to evaluate the compliance of a system or components with regards to performance requirements." Aptest went further to state that performance testing is often performed using an automated test tool to simulate large numbers of users, and thus it is also known as 'load testing'. According to Sommerville (1997), Splaine and Jaskiel (2001) performance testing can be grouped into the following categories:

- Smoke Testing
- (Work) Load Testing
- Stress Testing
- Spike/Bounce Testing

The results of these tests can be analysed and related back to the performance objectives of the Web sites. These different categories of performance testing will be discussed in detail below.

4.3.1 Smoke Testing

According to Aptest (2002) smoke testing is "a *quick-and-and dirty test which tests that the major functions of a piece of software works. It originated from hardware testing practise of turning on a new piece of hardware for the first time and considering it does not catch fire.*" Smoke tests are used to evaluate whether or not a software release is really ready for testing. Smoke tests can be as simple as performing a few manual functional tests with a simple modem and a stopwatch, or they can be as complex as running a fully automated load that reports on the approximate performance of the entire Web site (Splaine and Jaskiel, 2001).

The smoke test should exercise the entire system from end to end. It does not have to be exhaustive, but it should be capable of exposing major problems. The smoke test should be thorough enough that if the program passes, you can assume that it is stable enough to be tested more thoroughly.

4.3.2 Load Testing

Aptest.com (2002), a Web site dedicated to software testing, has a glossary where they define load testing. In the definition of load testing it is stated: *"see performance testing."* When one checks the definition of performance testing, it is defined as: "*testing conducted to evaluate the compliance of a system or component with regards to performance requirements. Often this is performed using an automated test tool to simulate large numbers of users. Also known as "Load Testing".'*

4.3.3 Stress Testing

Stress testing is designed to test the software with abnormal situations. This usually involves planning a series of tests where the load is steadily increased. Sommerville (1997) defined stress testing as the type of test which relies on stressing the system by going beyond its specified limits, and hence testing how well the system can cope with overload situations. According to Aptest (2002) stress testing is 'testing

conducted to evaluate a system or component at or beyond the limits of its stated requirements. Stress testing attempts to find the limits at which the system will fail through abnormal quantity or frequency of inputs. Examples of such inputs are:

- Higher rates of interrupts
- Data rates an order of magnitude above 'normal'
- Test cases that require maximum memory or other resources.
- Test cases that cause 'thrashing' in a virtual operating system.
- Test cases that cause excessive 'hunting' for data on disk systems.
- Can also attempt sensitivity testing to determine if particular combinations of otherwise normal inputs can cause improper processing.

Stress testing is particularly relevant for e-commerce Web based applications because these sites exhibit severe degradation when they are heavily loaded as the Internet becomes swamped with data which the different processes must exchange. Stress testing is used to determine if one's specific combination of hardware and software, which could be Web server, application server, database, network bandwidth, etc, has the capacity to handle an excessive large number of transactions during peak operating hours. Stress testing a Web site will also manifest what will happen when the maximum capacity is actually reached.

4.3.4 Spike/Bounce Testing

A good definition of spike/bounce testing was given by Splaine and Jaskiel (2001) where they wrote:

> *Spike testing of Web sites is analogous to predicting hurricanes. Weather forecasters are always wondering, "When will the storm hit? Where will it hit? How strong will it be?" Well, we already know the answer to one of these questions. The "storm" will hit your Web site, although some sites are more susceptible to these "storms" than others. This is due to the combination of predictable and unpredictable events throughout the world. Holidays, late-breaking news stories, special events, and product or service announcements can all cause spikes. The only question remaining is, when will the spike hit and how strong will it be?*

One possible variation of spike testing is to follow the spike with a low workload and then with another spike. Bouncing the load up and down will allow one to determine how well the Web site can handle the change in load and whether it is able to claim and release resources as needed. Spikes are characterised by random arrivals of client requests that significantly exceed normal averages. A Web site may experience load growth over a short period of time. While a Web site may be able to handle the load in a gradual ramp-up, the sudden nature of a spike can cause serious problems.

4.3.5 Analysis of the Testing Types of Endurance to Exceptions

In order to properly analyse the testing types discussed above the definitions will be restated briefly and any similarities noted.

- **Performance Testing**

  ...performance testing 'examines the ability to perform under high load.
  ... it is also known as 'load testing'...

- **Smoke Testing**

  ...can be as complex as running a fully automated load that reports on the approximate performance...

- **(Work) Load Testing**

  ...conducted to evaluate the compliance of a system or component with regards to performance requirements...

- **Stress Testing**

  ...testing how well the system can cope with overload situations..

- **Spike/Bounce Testing**

  ...Bouncing the load up and down will allow one to determine how well the Web site can handle the change in load...

The common words of the definitions of the above terms have been double-underlined to highlight the similarities amongst the different endurance testing types. It is evident that these endurance testing types and strategies are related and

interlinked in such a way that one would not have had conducted a proper performance test if smoke testing, load testing, stress testing and spike/bounce testing has not been also carried out. This is not unique to the Web sites to be tested. Ian Sommerville (1997) wrote that large systems are usually tested using a mixture of testing strategies rather than any single approach. It is therefore adequate to state that for the purpose of this study a mixture of the above strategies will be used.

4.4 Software Performance Testing and User Involvement

According to Shen (1998), the traditional attitude has held that involvement with users in software development is an evil to be avoided wherever possible; it interferes with the real job of programming and therefore significantly reduces productivity. From a software development perspective this is a very interesting statement. One would ask if this statement would hold true in the electronic environment where online shoppers want to access a Web site and do their shopping with ease and feel that the system does not complicate their shopping experience.

The emerging attitude asserts that building a system that users can put to productive use is at least as important as building a system productively. It is important to remember that with the electronic market users who experience problem with their online shopping experience tend to abandon the Web site and use another competitor's. With such facts, it can be concluded that significant user involvement, especially during the performance testing of the Web site, is mandatory. The problem is that software systems developed productively in isolation from users often fail to satisfy user's needs. This means that when there is a Web site to be tested for performance it is important that some research gets done to identify user performance expectations and requirements.

4.5 Performance Testing Assumptions, Methodology and Computer-based Testing

This section briefly introduces several notions necessary for the subsequent sections of the chapter.

4.5.1 Assumptions

Assumptions refer to what is considered as given for a particular testing activity. For software performance, the reference is often a representation of a typical software user's data. In the e-commerce environment a reference for a transaction processing system might be a benchmark data set representing a given transaction mix for a group of concurrent users, with performance stated in a form such as '300 transactions per minute with 255 concurrent users on the Web site.'

4.5.2 Testing Methodology

This is going to be discussed in more detail in chapter five, where different methodologies will be examined and analysed, and the one suitable for this study chosen and employed in conducting the experiments.

4.5.3 Computer-based Testing

This is also going to be examined in chapter five, where the main focus will be on automated testing. The tools used in computer based testing will also be examined and evaluated using a comparison criteria table. The evaluation will make it possible to choose a preferred automated testing tool to use for testing the applications.

4.6 Performance Testing and Performance Requirements

Glass (2001) had an interesting observation when he wrote that when addressing the issue of performance requirements (and not specifically testing) one needs to think about "back to front" approaches. He went further to write that to start performance testing at the back end, focusing especially on any legacy systems and their (baseline) performance, regarding performance requirements, the maximum response time is about eight seconds, and anything more than that requires telling the user what is going on.

What about the state of the Web applications practice? Hicken (2002) wrote that the average Web site has one bad link for every 3.5 pages and 12 errors for every page of HTML. He added that 90 percent of browser incompatibilities are caused by using non-standard HTML, and the other 10 percent are because the browser does non-standard things.' In the working environment it is common that Microsoft is blamed for a number of the failures because it is claimed that Microsoft deliberately does non-standard things so that their competitors could not get their code to work. It is important to remember that even if this could be true, the competitors themselves are also to blame for their own non-standard practices.

The problem with testing Web applications is that often enough one has to run tests without almost any requirements. When testing a Web site one has to know that there are requirements which need to be met. Conformance to explicitly stated performance requirements, which are explicitly and well-documented, can help produce a product which meets the standards of a professionally developed software.

Nixon (2000) wrote that managing performance requirements can be quite difficult due to the nature of performance requirements. Performance requirements can have a global impact on the target system. Meeting one such requirement may change several parts of a system. One cannot simply add a "performance module" to produce a system with good performance. Rather, one should consider performance requirements throughout the development process.

4.7 Performance, Functional and Usability Testing, and the Behaviour of Online Customers

The definition of performance testing has already been discussed above. In this section, performance will be examined for its relation to the functionality and usability testing, as well as how it (performance) affects and influence online customers. Functional testing and usability testing are not the focus of this study. Nevertheless, these two types of tests will be briefly overviewed for a number of reasons, which have a bearing effect in the performance of a Web site. This also has an effect in how long users can spend time navigating the Web site so that the performance measurement can reflect reliable figures. Customer behaviour is also examined because the performance of a site has a lot to do with whether or not customers return to a site, or get lost to other sites for good.

The issues that have to be addressed when doing performance testing differs in a number of ways from the issues that should be addressed when doing typical functionality and usability testing. Usability testing is one aspect of testing that is assessed by considering human factors, overall aesthetics and consistency of the Web site. Functionality testing is that aspect of testing that is assessed by evaluating the feature set and capabilities of the application, and the generality of the functions delivered (Louisa, 2002).

Functional testing is testing the features and operational behaviour of a product to ensure they correctly meet the specifications (ApTest, 2002). Another concept used to refer to functional testing is 'Black-box testing' (Sommerville, 1997). Black-box testing was examined in chapter two. It is testing done on an analysis of the specification of a piece of software without its internal workings. The goal is to test how well the component conforms to the published requirements (ApTest, 2002).

Palmer (2002) wrote that Web site outcomes can be influenced by a number of usability and elements. According to Wikipedia (2002) usability testing is a means for measuring how well people can actually use something for its intended purpose. Usability testing is a measure of whether potential users of the Web site are able to

find what they need with reasonable effort and within reasonable period of time. Usability also encompasses how easy it is for new or infrequent users to learn to use the Web site (Splaine and Jaskiel, 2001). ApTest (2002) defined usability testing as testing the ease with which users can learn and use a product. Splaine and Jaskiel (2001) also stated that there are two types of Web usability testing: site-level (system) testing and page-level (unit) testing. Site level usability testing is concerned with information architecture, navigation, page templates, site layout, consistent graphics, linking strategy, and search capabilities. Page-level usability testing pertains to specific issues and problems that occur on individual Web pages. Marick (2002) wrote that expert customers often do not report usability problems, because they have been "trained" to know it is not worth their time. Instead, they wait (in vain, perhaps) for a more usable product and switch to it. Testers can prevent that lost revenue. Palmer (2002) wrote that Web site owners need to consider usability and other design criteria so that they could be successful, and at the same time satisfy their users.

When testing a Web site for its performance, especially with stress testing and load testing, particular attention should be paid to the functional integrity of the Web site while it is being stressed in order to ensure that the functionality that worked correctly during periods of low load still works correctly while the site is being stressed.

The growth of Internet usage, spurred by rapidly advancing network technologies, has been bringing about broad changes in customer behaviours related to the Internet, attitude towards mass media, and satisfaction (Cho *et al*, 2003). These writers also stated that Internet behaviours refer to purchasing over the Internet. It has already been noted that online buyers abandon online purchases because of a number of reasons. It was also noted that most of the times the number one reason is poor performance of the online shops which load at a very slow pace to an extent that the users get frustrated of waiting. According to Dittmeier (2000) one second can cost an organisation thousand of rands. The Internet brought about a whole new kind of user. This user typically has very little loyalty to a Web site. The user will not even stay long enough to use the Web site if it is not immediately clear how the site should be used without any training whatsoever.

The biggest mistake that any e-commerce Web site could make is to try to mould its customers into doing business the way the company wants to do business. Unlike traditional in-house end-users, Internet users are not "captured" and are therefore free to choose to view or use another company's Web site. This can be significantly increased by a Web site which is not useable as well as lacking functionality. A study by Zona Research (2002) found that sixty-two percent (62%) of Web shoppers had abandoned their attempts to find and purchase items online. Approximately twenty percent (20%) of these Web shoppers had quit more than three times during a two-month period. Cho *et al* (2003) concluded that waiting time on the Internet has a negative influence on the consumers' evaluation of Web sites. It is therefore important to ascertain that the Web site is both useable and functional before performance testing can be conducted. This can help in making sure that by the time performance testing is carried out the reports about users abandoning the Web site due to performance is a true reflection, and not usability and functionality problems confused and mistaken for performance problems.

This above discussion illustrates that it is important that for the system to be tested for performance it should also be tested for usability as well as functionality. It would be a worthless exercise to test a system for performance and ignore usability and functionality because, according to Bacheldor (1999), the growing emphasis in building high-performance, bulletproof, customer-friendly Web sites has revealed the soft underbelly of e-commerce: a lack of tools for managing it all. Scalable servers make it possible for businesses to build sites that support thousands of visitors, but a dearth of sophisticated add-on tools has made it difficult to assess system performance and understand and react to customer behaviour. Online customers should have a positive experience on a business Web site because it reflects on the company.

4.8 Performance Testing Objectives and Goals

Before embarking on a performance testing activity one should identify the performance fail/pass criteria to assess what performance testing objectives have been set (Splaine and Jaskiel, 2001). Often testing is started without first having a clear understanding of what should be tested.

An important issue to remember when doing performance testing is scalability: the ability of the system to handle significant heavier workloads than are currently required (Weyuker and Vokolos, 2000). This necessity might be due to such things as an increase in customer base or an increase in the system's functionality. Either of these two changes would typically cause the system to have to be able to provide significantly increased throughput. If there has not been appropriate testing to assure that the system can be scaled, unacceptable response times might occur as workloads increase.

CPU failures, bus failures, and memory errors were identified in chapter three as the reasons for system crashes. This normally causes system downtime.

There are a number of goals which could be set when designing a research project related to performance testing. According to Gartner (2001) the following are the goals of performance testing:

- determine how many users can the system manage concurrently
- determine how long it take to restart an application on another system
- determine if additions to the system can be made while the system is still operational
- find out if a fail-over process is transparent to the system users

Weyuker and Vokolos (2000) listed several performance testing objectives. These are:

1. the design of test case selection or generation strategies specifically intended to test for performance criteria rather than functional correctness criteria,

2. the definition of metrics to assess the comprehensiveness of a performance test case selection algorithm relative to a given program,
3. the definition of metrics to compare the effectiveness of different performance testing strategies relative to a given program
4. the definition of relation to compare the relative effectiveness of different performance testing strategies in general; this requires that one be able to say in some concrete way what it means for performance testing strategy X to be better than testing strategy Y,
5. the comparison of different hardware platforms or architecture for a given application.

For the purposes of this study, the first and third objectives were considered. Test cases are needed to run the experiments, which will be planned and designed in the next chapter. A number of testing strategies will also be examined to find one, which could be suitable for the purposes of this study.

Cohen *et al* (1997) wrote that testing is an important but expensive part of the software development process, and much research has been aimed at reducing its costs. Taking this statement into account, one should exercise extra caution and care not to try to reduce testing costs at the expense of compromising best testing practises.

Nixon (2000) called for performance goals to be specific and measurable. Jain (1991) provided the acronym "SMART" as a reminder that performance requirements should be specific, measurable, acceptable, reliable, and thorough. Concerning specificity of goals, peak times and off-peak times will be distinguished.

4.9 Performance Testing Approaches and Environments

Software exists in an environment in which other entities (users) stimulate it with inputs. The software provides those users with output. A tester's task is to stimulate interaction between software and its environment (Whittaker, 2000). Testers must identify and stimulate the interfaces that a software uses and enumerate the inputs that can cross each interface. This might be the most fundamental issue that testers face, and it can be difficult, considering the various file formats, communication protocols, and third-party availability. According to Munson (1992), too many performance and reliability models treat programs like black boxes whose internal attributes do not differ from one application to another. Even though this statement was made over ten years ago, when the performance of Web sites was not much of an issue, it is interesting to note that it is very much observed today. The problem is that the Internet and the Web environment have presented the software engineering community with new challenges of dealing with software performance.

When visiting most Web sites today, one is bound to be using more than one server software application package. This is especially true for dynamic Web sites offering personalised content. The information changes every time as the database system gets updated, and such changes reflect on the Web sites dynamically. Behind the scenes, teams of engineers are doing their best to keep the back-end systems interoperating. Software engineers are constantly writing expensive and difficult-to-maintain custom software to keep the system sharing data (Cohen, 2002). In chapter two the relation between software performance and software maintenance was established. If such complex systems are developed and are difficult to maintain one can assume that the same applies to performance testing, that it is also difficult to carry out on such systems.

4.10 Chapter Conclusion

In this chapter a number of issues relating to application performance testing were investigated. These ranged from the challenges faced by organisations that conduct their business online, to behaviour of the online customers. It was also noted that for an organisation to make it in the Web environment, performance of the Web site should be at acceptable levels. It was also noted that the online customer is a new type of a customer who typically has very little loyalty to a Web site.

Because online shoppers do not tolerate a poorly performing site, they quickly click away to another competitor's site without any hesitation. This has an effect of organisations losing business on the Web. This translates to a need for proper testing of e-commerce Web-based applications. The next chapter will focus on planning for such tests.

Chapter 5: Test Planning Issues and the Design of the Experiments

5.1 Introduction

Testing should be planned and conducted systematically (Louisa, 2002). Often testing is started without first having a clear understanding of what should be tested (Splaine and Jaskiel, 2001). According to Marick (2002) one of the classic testing mistakes is 'testers paying more attention to running tests than to designing them.' A tester who is not systematic, who does not spend time laying out the possibilities in advance, will overlook special cases. Sabin (1998) wrote that planning for the testing process is the key to successful completion of client-server product performance testing. Except for the simplest of Web sites, it is apparent that some sort of planning is needed (Splaine *et al*, 2001). This chapter is essentially on the test planning activity. The type of application, system and environment determines how one designs and develops testing plans. The objective of system test planning is to thoroughly understand the nature of the application to be tested, through a systematic process, to provide a clear, comprehensive road map for efficient application/system testing (Systest, 2001).

This chapter explores, in detail, the testing plan design and formulation of Web site performance testing experiments. These are based on the theoretical foundations (to be discussed later in this chapter) and the scope of this research (which has already been addressed in chapter 1). This chapter has three main focuses; firstly, it explores the theory applicable to software testing plan design, with a main focus on Web site application testing. Secondly, a methodology for testing the Web applications is examined. Thirdly, it presents the evaluation and selection of an automated testing tool to employ in conducting the tests, as well as the testing plan, test procedures, test cases, and the preparations of the automated tool which was used in testing the Web sites.

The most commonly used approach to obtain performance results of a given Web service is *performance testing*, which means to run tests to determine the performance of the service under specific application and workload conditions. It is crucial to

predict how a particular Web service will respond to specific workload. This can only be achieved by properly planning the tests and executing them in a way that will provide results which could be analysed, and conclusions drawn on the system's performance. This chapter studies such methods which can be adopted in planning, selecting, executing and documenting test sets for Web performance testing that satisfy criteria for execution paths and input domains. The main emphasis in this chapter is on researching the test planning processes and selecting the best approach to employ for the purpose of this study. The next chapter handles the running and documentation of the tests.

It is important to examine the reasons why it is necessary to plan test activities before embarking on the testing process. According to Yamaura (1998), documented tests can be beneficial in several key ways, namely:

- designing test cases gives you a chance to analyse the specifications from a different angle;
- you can repeat the same test cases;
- somebody else can execute the test cases for you;
- you can validate easily the quality of the test cases;
- you can easily estimate the quality of the target software early on.

According to Larson (1995) test cases are single testable instances with either a right or wrong answer. Test cases provide another rendition of the functional specification. Designing the test cases will give the tester a pause to think of the conditions which could not be considered (Yamaura, 1998). Many organisations are seeking to deploy mission-critical Web sites and applications that are intended to attract large numbers of visitors, often with significant revenue associated with the activity (Splaine and Jaskiel, 2000). Such applications need to be developed once they have passed well planned and properly executed tests.

One of the often neglected, but extremely important parts of the software development process involves testing the product to ensure that the developers made no obvious errors. When done correctly, this entails developing a strategy for testing the software, preparing a plan for implementing the strategy, actually running the

software in an attempt to detect errors, and recording the results of the tests so they can be recreated if necessary (Harrison, 2000).

The quality of systems developed and maintained in most organisations is poorly understood and below standard. This chapter examines ways of how testing can be planned and executed effectively in an effort to curtail the problems associated with software which has not been tested properly. In line with this, Sabin (1998) proposed the following activities to be included in the test definition phase of performance testing:

- identify the key performance goals;
- defining performance pass/fail criteria;
- identifying the measurement criteria as defined, measurable metrics that can be successfully captured and analysed;
- carefully analysing the product and its expected operational use;
- identifying the key transactions and transaction mixes required;
- identifying the test environment(s) required;
- identifying the best balance between physical client execution and virtual client or virtual database user simulation;
- analysing various test approaches and selecting the best one for this effort;
- selecting the best test automation tool or combination of tools for test script development and execution;
- clearly transforming this information into performance metrics than can be easily understood and analysed from the performance testing results;
- obtaining consensus on the acceptability of the identified goals, test approach and test criteria.

Therefore the work in this chapter aims to find out those that are applicable and relevant for this study. The next section will explore some issues related to the facilitation of the test experiments.

5.2 Background Issues to the Experiment Facilitation

The effects of poor performance on the profit of business organisations have already been examined. The loss of existing and potential customers cannot be tolerated. Every business organisation running online shopping services needs to know how its e-business system stands up to market demands. To find out the answers to such concerns, the system testers can set up a test environment, simulating thousands or tens of thousands of virtual users and conducting load and performance test (Systest, 2001). To do this, one needs the appropriate hardware, automated test tools and the required expertise. To carry out the testing activities there must be an individual responsible for the overall planning and design activities. In business organisations this could be the task of a software project manager or project leader. In this study the author will be carrying out all these tasks, and will be hence called the facilitator.

The Experiment Facilitator

The facilitator will conduct and supervise the test experiments for this study. According to Weyuker *et al* (2000), core competencies of a software testing facilitator consist of general software engineering skills and knowledge. Some of these competencies can be acquired during study for an undergraduate or Master's degree and others during the early years of working in corporate environment. Such skills include general topics in telecommunications, basics in networking, and technical analysis skills. A software tester must be thoroughly familiar with the functionality, performance, and usability of currently available test tools. Test planners must understand the development methods and environment to effectively plan for testing (Klaasedev, 2002). Marick (2002) wrote that a good tester has the following qualities:

- methodical and systematic;
- tactful and diplomatic (but firm when necessary);
- sceptical, especially about assumptions, and wants to see concrete evidence;
- able to notice and pursue odd details;
- good written and verbal skills (for explaining bugs clearly and concisely);
- a knack for anticipating what others are likely to misunderstand;
- a willingness to get one's hands dirty, to experiment, to try something to see what happens.

Equipped with experience from having worked as an account executive, project manager, technical team leader, business analyst, technical systems analyst (all in a Web environment) and lecturing at departments of information systems and technology, the facilitator of these test experiments meets the above mentioned basic preferred competences and skills for being a software tester, and will be able to conduct the test experiments of this study's magnitude.

5.3 Theoretical Foundations of the Tests and Experiments Design for this Research

5.3.1 Discussions on the Different Test Plans Explored for Consideration in this Study

(a) The Testing Process

When working on a project, one needs to formulate an overall testing plan during the analysis and planning phase. The testing plan defines the overall approach to testing and describes how the testing process will ensure that the solution has the appropriate level of quality and reliability. The testing plan provides the overall guidelines from which all testing decisions are based. A well-crafted testing plan allows the rest of the testing process to be defined more effectively. The testing plan needs to be understood and approved. If the plan is accepted, there is a much better likelihood that the final solution will meet the customer's expectations and those of other stakeholders.

(b) The Testing Plan

A good testing plan is a cornerstone of a successful testing implementation (Software Testing Institute, 2002). According to Hower (2002), a software project testing plan is a document that describes the objectives, scope, approach, and focus of a software testing effort. A testing plan is a document that answers the basic questions about one's testing effort (Thehathawat, 2002).

The process of preparing a testing plan is a useful way to think through the effort needed to validate the acceptability of a software product. According to Marick

(2002) a test design should contain a description of the setup, inputs given to product, and a description of expected results. The items which can be included in a testing plan will be discussed below. Bach (2002) defined a testing plan as a set of ideas that guides the test process. The purpose of the testing plan in a testing event is the translation of the testing strategy to lower-level detail (Mochal, 2001).

There have been a number of testing plans suggested by various writers. Some of these will be discussed in this section, and the one (testing plan) to be employed in this study will be chosen based on its closeness to what this study purports to achieve. Not all practices are widely known or greatly documented, but they all possess the strength that is powerful when judiciously applied (Chillarege, 1999). Thus, this study will adopt the plan that will enable it to achieve its goals. The following discussion is on several testing plans proposed in the literature.

❖ Testing Plan by Grossman

According to Grossman *et al* (1996) the testing plan must include procedures that document the steps involved in executing a test, including:

- backup and recovery procedures to return to the baseline data before and after tests;
- test execution procedures;
- test logging; and
- test result documentation.

❖ Testing Plan by Hagen

According to Hagen (1998) a testing plan should define:

- the scope and objectives of the project;
- any assumptions that may be underlying the project;
- resources needed (both human and non-human);
- the requirements and the criteria against which they are to be tested;
- a detailed testing schedule;
- test case design;
- test scenario design.

❖ Generic Test Generation Procedure by Murray and Hayes

Murray and Hayes (1996) suggested the following test generation procedure, which according to them is generic to all Test Generation procedures.

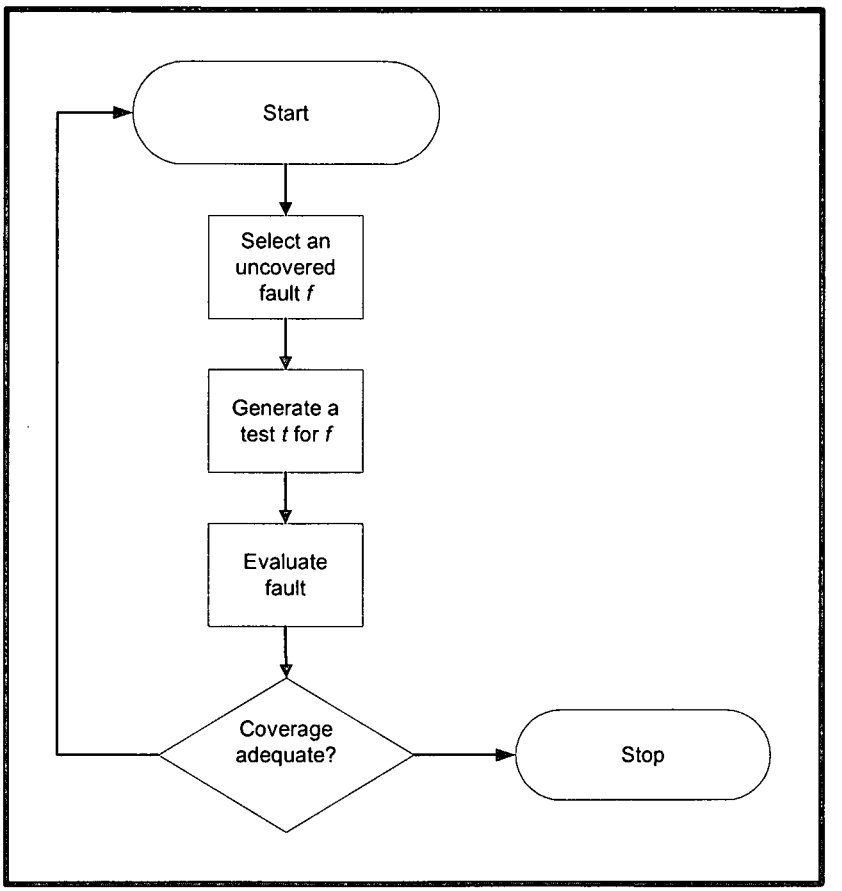

Figure 5.1: Generic Test Generation Procedure (Murray and Hayes, 1996)

Figure 5.1 outlines a generic test procedure. The test-generation step in the middle of the figure has two basic parts:

1. activate, or expose, the selected fault; and
2. propagate an error signal from the fault site to an observation output

This procedure assumes that a system fails when the service it provides differs from the service it was designed to provide. A system error is when its state differs from the correct state. A fault is the physical difference between a good system and a failing one. The goal of testing is to detect faults and thus prevent failures.

According to Hower (2002) the steps to software testing are:

- obtain requirements;
- determine project-related personnel and their responsibilities, and processes;
- identify application's higher-risk aspects, set priorities, and determine scope and limitations of test;
- determine test approaches and methods – unit, functional, load tests, etc;
- determine test environment requirements (hardware, software, etc);
- determine test requirements (record/playback tools, test tracking, etc);
- determine test input;
- identify tasks, those responsible for tasks, and labour requirements;
- set schedule estimates, timelines and milestones;
- prepare test plan documents and have needed review/approval;
- write test cases;
- have needed review inspections/approvals of test cases;
- prepare test environment and testware, etc;
- obtain and install software releases;
- perform tests;
- evaluate and report results; and
- track problems/bugs and fixes.

These steps seem to be very close to what this study aims to achieve because most of the activities are relevant for this research. Nevertheless, activities like bug fixes will not be carried out in this research because it falls outside the scope of this study.

❖ Performance Testing Program by Nguyen

Nguyen (2000) stated the following considerations for testing performance on the Web:

- determine which tool(s) are most appropriate for the test project;
- determine the performance factors and requirements;
- determine a projected number of user;
- run performance testing as many times as possible;
- identify server-side analysis of performance bottlenecks;
- determine acceptable performance levels; and
- do a data analysis and corrective action planning of how performance degradation can be resolved.

❖ Performance Testing Steps by Empirix

A report by Empirix (2003), a company specialising in testing Web applications, listed the following, as the critical steps that one should take when planning for performance testing:

- **strategy** – lay out the strategy one will use for testing the application;
- **objectives** – document and get buy-in from other relevant parties on the objectives of the load testing;
- **scope** – clearly define the scope by determining what will and will not be tested during the testing;
- **performance requirements** – document the system performance requirements, which are used to determine whether the system passes or fails;
- **test cases** – document the test cases and scripts;
- **resources** – document what resources (system and personnel) will be needed;
- **tools needed** – document all tools that will be used during the testing and what role each will play;
- **failure strategy** – in case serious errors or performance problems are found during testing, have a plan in place with buy-in from all the stakeholders on how those problems will be addressed and what type of resources each party is willing to dedicate to resolve them.

5.3.2 Analysis of the Test Plans from the Previous Section

All these testing plans illustrate that while every testing effort may be unique, most testing plans include a common content framework. These testing plans are good in their own regard, but a close analyses of these testing plans reveals that the ones by Nguyen (2000) and Emprix (2003) very closely suit the requirements of this research. These plans (Nguyen's, 2000; and Empirix's, 2003) were both proposed for performance testing of Web-based applications. Nguyen proposed his plan in his book which is titled '*Testing Applications On The Web: Test Planning for Internet-based Systems*' while Empirix proposed theirs in their article titled '*Top 25 Reasons Web Applications Don't Scale'*. Thus, therefore, a fusion of the two plans is considered appropriate for this research, and will be applied.

The fact that some of the other testing plans did not perfectly meet the requirements of this project is not surprising. Most of them were proposed for applications which were not Web-based, and the Web technologies have presented the software engineering community with new challenges. The other testing plans were not chosen for their disadvantages which entail:

- being expensive to create;
- the inevitable minor changes to user interface will break it, which basically means that it is more expensive to maintain;
- when one makes a minor mistake the whole process has to be aborted and restarted;

Very little literature exists on Web technologies testing. Currently there are a number of research projects being conducted (Elbaum, 2003). It is hoped that in the near future there will be more sources on Web technologies related material for use in future research projects. The *'action plan for this experiment'* section, which is examined later in this chapter, re-examined the preferred testing plans and stated, expressly, the exact actions which were carried out during the experiments.

In addition to the testing plans discussed above, the following activities also need to be carried out for this research:

- backup and recovery procedures to be able to return to the baseline before the tests;

- test logging;
- test result documentation;
- assumptions preparations; and
- start and stop test when coverage is adequate.

The above testing plans provide clear steps into the procedures of testing software. In following such plans, one needs to also note those things which should be avoided. The next section examines such things in the form of mistakes which had to be avoided.

5.3.3 Potential Mistakes which were Identified and Avoided in this Study

Marick (2002) stated some classic testing mistakes, which were also considered and avoided in this study. He clustered these mistakes into five groups, namely:

- the tester at work: designing, writing, and maintaining individual tests;
- technology rampant: quick technology fixes for hard problems;
- being too specific about test inputs;
- ignoring the economic propositions of test automation, and automate all test, even those that are not going to run often enough to justify it;
- testing for performance without clearing resources like dynamic routers, etc.

Every effort was made to avoid such mistakes in this research. Furthermore, a tester should also avoid paying more attention to running tests than to designing them. Other mistakes to avoid include failing to also conduct tests for those tests which are closely related to the test being conducted. In this study it would make no sense to only test the Web applications and totally ignore functionality and usability testing. The necessity of testing for functionality and usability has already been discussed above. Usability and functionality testing are both going to be carried out because if they were ignored then the project would be defining the testing task too narrowly (Marick, 2002). Testers are restricted to checking whether the product does what was intended, not whether what was intended is useful. Customers do not care about the distinction, and testers should not act like customers.

5.3.4 Resources Needed in Conducting the Experiments

It has already been stated above that most of the parts of the different testing plans were not considered because they fall out of scope. Some of these plans listed 'resources' as one of the items required in testing. For example, Hagen (1998) stated that, there is a need to state the resources needed for the test. The human resources needed are:

- the *facilitator* – responsible for facilitating the whole testing procedures and processes;
- some real *end-users* – one might need real users to be able to gather some information which might not be obtained if only virtual users were employed;
- software in a form of *Automated testing tool* will be required for use in testing the applications;
- hardware in a form of *servers* or *controllers* where the testing software will be installed, executed and monitored;

Splaine *et al* (2001) wrote that 'the easiest way to plan (or design) the testing of a Web site/application is to develop a series of testing plans. These plans help ensure that the Web site/application is comprehensively tested and can be reliably re-tested. Each testing plan is focused on a different phase of the testing process, which could be unit testing, system testing, or post-implementation testing, while the individual test cases within each testing plan are grouped into categories that focus on a single aspect of testing, which could be usability, compatibility, performance, etc.' In the case of this study, only performance (related) tests are conducted, which means that there is no need for a series of testing plans to be prepared as per this suggestion. What needs to be suggested next is the methodology, strategy and procedures for testing.

5.3.5 Testing Best Practices

Not all practices are widely known or greatly documented, but they all possess the strength that is powerful when judiciously applied. According to Chillarege (1999), practices can be described by grouping them under *Basics*, *Foundational*, and *Incremental*. These are stated in table 5.1 below.

Table 5.1: Testing Best Practices (Chillarege, 1999)

Basic Practices	Foundational Practices	Incremental Practices
▪ Functional specifications ▪ Review and inspection ▪ Formal entry and exit criteria ▪ Functional test - variations ▪ Multi-platform testing ▪ Internal betas ▪ Automated test execution ▪ Beta programs ▪ 'Nightly' builds	▪ Use scenarios ▪ Usability testing ▪ In-process feedback loops ▪ Requirements for test planning ▪ Automated test generation	▪ Teaming testers with developers ▪ Automated environment generator ▪ Testing to help ship on demand ▪ Memory resource failure simulation ▪ Statistical testing ▪ Check-in test for code ▪ Benchmark trends

These practices will be used as subsets of the already mentioned testing plans. Those that fall out of the scope of this research will not be used.

5.3.6 Designing Test Cases and Test Data

There are several strategies that could be used to successfully acquire test data. These are:

- generate the data by hand. That is, manually 'cut and paste' data into the input files or directly type the data into a browser;
- use existing Web logs as a starting point and then manually 'massage' the data to fit the test requirements; and
- use a tool to capture an initial template and then randomise the data.

According to Yamaura (1998), there is only one rule to designing test cases: cover all the features, but do not make too many test cases. To find the bugs that customers see, that are important to customers, one needs to write tests that mimic typical user tasks. This type of testing is called scenario testing, test-based testing, or use-case testing (Marick, 2002). Data can also be collected during the testing processes. Data collected as testing is conducted provides a good indication of software reliability and, to some extent, quality (Hamptonu, 2002). If the test cases are properly developed, one can easily estimate the quality and performance of the target software in the midst of test and debug.

During the process of designing the test cases, one needs to be careful not to design too few or too many test cases. As Yamaura (1998) stated, too few test cases will leave easy bugs undetected, and too many will run short of time. Yamaura (1998) went further to suggest the following six core procedures for systematic testing, which focuses mainly on test cases:

1. design a set of test cases.
2. check the test cases.
3. do code inspection based on the test cases.
4. do machine debugging based on the test cases
5. collect quality-related data during debugging
6. analyse the data during debugging

According to the goals of this project, only points 1 and 2 are appropriate for this research. The other points fall outside the scope of this study. The issue of *garbage in garbage out (GIGO in short)* is well known in the computing discipline. It follows,

therefore, that the test cases used in this study also need to be validated in order to avoided getting 'garbage' results. The next section examined the validation of the test cases.

5.3.7 Validating Quality of Test Cases

Test cases must be tested. This is an important activity because it helps in ensuring that they cover all the features implemented in the software. When the test case design is completed their validity and correctness will be evaluated based on:

- whether the test cases cover all scenarios;
- the balance between normal, abnormal, boundary, and environment test cases;
- the balance between black-box and white-box testing;
- the balance between functional tests and performance tests; and
- their overall sufficiency.

5.4 Testing Methodology, Strategy and Procedure

In order to ensure that the tests are reflective of the customers' experiences, field tests will be conducted from remote end-user's connections. Fieldwork will be conducted in order to fall in line with Systest's (2001) statement that one's target Internet audience will not access one's Web site from one big room (laboratory) with a certain number of connections. It is important to note that some Web site performance is usually carried out in small, isolated local area networks, with almost no transmission errors. This was also confirmed by Chevalier and Quarterman (2002) in writing that "a few companies in the field of software testing focus on localised performance measurement reports for network, rather than monitor the global network to see how it may affect their bit of the world." While some companies claim to measure Internet performance, they usually mean Web site availability and traffic within their own networks. The presumption that permeates this notion is that the company's network is isolated from the rest of the Internet, and that as long as their routes remain viable, the rest does not matter. One needs to realise that Web services, offered in "real world" are accessed through the Internet or large Intranets, which involve wide area

network connections, gateways, routers, bridges, and hubs that make the network environment "noisy" and error-prone. Moreover, latencies are much higher in real world environment than in the local area network used in benchmarking efforts.

Chevalier and Quarterman (2002) wrote that downed routers in Tokyo may directly affect customers of an e-commerce system in San Francisco. Interdependence forms the basic notion behind the Internet and as such understanding the performance of the entire system directly informs local network well-being. But the double-edged sword of the Internet also means that data and information can be dynamically re-routed successfully, if other arteries in the network are available. The test and analysis of a Web site and its application should take this observation into account. Hence field work for this research is justified. In this study, this will involve conducting the test experiments away from the company's network.

Depending on the project, there may be other high-level sections to include, such as testing objectives, testing assumptions, testing organization, and testing tools, along with effort and cost estimates, (Mochal, 2001). Such items as testing tools, testing assumptions and testing objectives will be included in this study. For instance, automated testing tools will be evaluated and assumptions will be made for the tests.

A testing methodology comprises of test strategies, test plans, test cases and test scripts, (Thehathaway, 2002). Different authors have written on the theory in testing plans, with some of them using different terms to refer to the same thing. Some writers talk about test procedures while others talk about test strategies. The interesting thing is that when one examines the content of these categories one discovers that it is almost the same. Nevertheless, this study will adopt one or a mixture of strategies which try to give the two sides and adopt the best one which can be useful for the purpose of this study.

The content that goes into the strategy is information which one will need to know at some point in the project. The question is whether one plans early or waits so long that you have to rush through the testing decisions at the back end of the project. As already noted, one has to plan the testing activity thoroughly before execution. After careful consideration of design styles, one can outline a comprehensive testing

procedure and methodology to be used as a guideline for conducting the tests and experiments (Miller, 2000).

5.4.1 Methodology

In order to achieve the goals of this study the first thing needed was to find a suitable real number of participants to serve as the real users of the study; then make use of them in the experiments. The extra/additional numbers of users would then be the virtual users generated by the Web automated testing tool. Factors that were deemed important were:

- active real user involvement in the experiments;
- the amount of real users contributed content or "knowledge-base" that the users have; and
- total number of user capacity (including the virtual users).

Secondly, there was a need to get feedback from users, and find out how they felt with their Web shopping experience. This was done by pooling together the real users and the virtual users, thereby making them part of one community of the testing participants.

Thirdly, to get test data, several readings were made on the Web statistics using Web Trends. Those areas of the sites visited most were picked as the test areas. From a researcher's perspective, using the most visited parts of the Web sites was a great advantage because this meant that the users would be interacting with the most requested services of the Web sites.

5.4.2 Description of what was Included in the Testing Strategy

The purpose of the testing strategy was to define the overall context for the entire testing process. The process was different depending on the specific characteristics of the solution. In many respects, this was the most important part of the testing process, since all testing decisions were made within the context of the strategy. According to Mochal (2001), the basic parts of the testing strategy are:

- **project overview**: Which is copied from the Project Definition.

- **business risks**: These are high-level risks of the project that will affect the overall testing strategy. This can include risks such as doing business on the Internet, which may drive the need for rigorous system tests of firewalls, technical architecture, and security. The risks can be classified as high, medium, and low, depending on the nature and impact of the problem. For each high and medium risk, one should identify what elements in the overall testing approach will help in ensuring that the potential problem does not occur.

- **testing milestones**: This section gives a preliminary overview of the testing timelines. Obviously, since this document is created in the analysis phase, these dates are subject to later revision.

- **testing approach**: This describes the testing process at a high level, including how one will conduct unit testing, integration testing, system testing, performance testing, and acceptance testing. If the project is large enough, each of these might be its own section. This is where one makes fundamental decisions regarding the type of testing that makes sense for the project.

- **testing environment**: One needs to think through the technologies and facilities needed for the testing process. If the overall testing environment needs are understood up front, it will be easier to break out the specific activities required to put the environment in place. In addition, one may need to plan for and acquire some parts of the environment well in advance.

According to Parkin *et al* (2002), the best testing environment is that in which the application is going to be implemented and operated. This maximises the likelihood of identifying problems. This study stands to benefit more in that the Web sites will be tested on their operational environment.

- **testing requirements:** deciding on what to test and how best to test an application is a crucial part of an effective testing process. A test requirement is a formalised structure to hold this information, (Koloski, 2003). One of the roles of software testing is to ensure that the product meets the requirements as originally specified in the requirements document. Capturing the requirements therefore becomes an essential part not only to help develop but also to create testing plans that can be

used to gauge if the developed product is likely to meet customer needs (Chillarege, 1999). This makes the management of requirements and its translation into testing plans an important step.

5.4.3 Users Participation in the Experiments

The automated testing tool, which was used to conduct the test experiments, made use of virtual users only. Unlike real/physical system users, virtual users are not capable of expressing what they experienced and felt when using an application. Thus, for the purpose of this study, real/physical system users were involved in the performance testing. The reason for doing this is in accordance with the statement by Larson (1995) where he wrote that 'end-users provide feedback on the effectiveness of the system – they indicate where more testing is required.' Poston and Sexton (1992), also wrote that test quality is important, with the customer being the ultimate judge. According to Potosnak (1998), the users in a test should be representative of the intended user population. As already stated, the tools which were used to conduct the tests for this research used virtual users. It follows; therefore, that Potosnak's point, and those of the other noted writers, should be followed in this study.

Another important consideration is the inclusion of the users in the testing plan. The testing strategy should be written at a high level so that it can be read and understood by the customer. Since all testing decisions are based on the strategy, the major stakeholders should approve the same document in order for it to be a success.

Linberg (2000) wrote that people act on the basis of what they perceive the situation to be, whether the perceptions are accurate or grossly inaccurate. Since behaviour is based on perceptions, the existence of each of them is a fact to be considered. Similarly, the frustrations, attitudes, loyalties, and hostilities felt by each user member and the information and misinformation possessed by each are facts, as is their evaluation of the merits and desirability of each particular course of action under consideration. A study on real users of e-commerce Web sites is not the focal point of this research. It is only done here for auxiliary exploration purposes. As already stated, the behaviour of real users has a bearing on whether the Web site is continually used or not. It is important to make good out of the testing plan and the applicable

strategy because ultimately one will have to live with the resulting testing process, and if it does not cover the users of the system, success can never be ensured.

5.5 Issues on the Use of Manual and Automated Testing

5.5.1 Manual Testing

Most testing, analysis, and debugging work is ad hoc and done manually. Humans are relied on to use their intuition and judgment to create test cases, evaluate results, and decide when testing is sufficient (Osterwell and Clarke, 1992). In a number of times customers expect Web sites to fail, often during the days and weeks after implementation. Many Web sites never satisfy their customers. Some of these sites are abandoned, and thus they should be rebuilt to meet customer needs, and some just go into the statistics of those sites that failed to take off because of lack of new funding. In every case of 'manual testing' decisions about testing, evaluating, correcting, and deployment rest almost exclusively on unsupported, sometimes haphazard, human judgment. Success stories on manual testing is largely attributed to human effort and dedication. According to Osterwell and Clarke (1992), tools to help analyse and test software could significantly reduce labour and demonstrably effective measures of test adequacy and software reliability could reduce reliance on fallible human judgment. This statement advocates the use of automated testing tools in testing. Automated testing is examined below.

5.5.2 Automated Testing

Automated testing is testing that employs software tools which execute tests without manual intervention. It is normally applied in graphical user interface (GUI), performance testing, etc, (ApTest, 2002). The growth of automated testing capability has stemmed in large part from the growing popularity of rapid application development (RAD), a software development methodology that focuses on minimizing the development schedule while providing frequent, incremental software builds, (Dustin *et al*, 1999). Client/Server and Internet computing systems require rigorous testing, but they present challenges which go far beyond those of traditional systems testing, (Mosley, 1999). The goal of automated test execution is that one

minimizes the amount of manual work involved in the test execution, and gains higher coverage with a larger number of test cases, (Chillarege, 1999). Simply put, 'test automation' is automating the manual testing processes currently in use, (Test Focus, 2002). The automated test execution has a significant impact on both the tool sets for test execution and also the way tests are designed. Integral to automated testing environments is the test oracle that verifies current operation and logs failure with diagnosis information. According to Systest (2001) when one automates the software testing process, one makes an investment that pays off in the future software test program. Each future test cycle will take less time and require less human intervention. In addition, one will have more confidence in the quality of the application because of the increased test effort and test coverage made possible by automated software testing.

Assuring Web site quality requires conducting sets of tests, automatically and repeatably, that demonstrate required properties and behaviours, (Miller, 2002). In line with this statement, this study employed automated testing tools and ran a number of tests that produced Web application performance results. Software test automation should not be taken lightly as if one can simply buy one of the popular test execution tools and simply do like one of those plug and play game applications. Graham and Fewster (1999) wrote that just as there is more to software design than knowing a programming language, there is more to automating testing than knowing a testing tool."

Today many people are using the Internet for a number of reasons. These people use a number of different Web browsers which they themselves are running on a number of different platforms. When code is ported from one platform to another, modifications are sometimes done for performance purposes. The net result is that testing on multiple platforms has become a necessity for most products. Using manual testing would make it hard and tedious to achieve test results which would cover most of the popular platforms while, on the other hand, test automation makes testing Web site/application performance achievable in a week or two.

The choice of using automated testing in this study is based on the simple economic proposition by Marick (2002). This economic proposition states that:

- if a manual test costs RX (in Rand) to run the first time, it will cost just about RX to run each time thereafter, whereas:
- if an automated test cost RY to create, it will cost almost nothing to run from then on.

Further on, the employment of automated testing in this study is justified because stress or load tests may be almost impossible to implement manually. It would be difficult to execute and check a function a thousand times. It would also be difficult to sit down hundred or more people at hundred terminals to have them running concurrent use of the application. According to Laursen (1998), automated testing can allow one to cover more ground in a shorter period of time than manual testing, and can assist in simulating load and activity. In a Test Focus (2002) article, the following was noted as the benefits of test automation:

- test can be run faster;
- run more tests more often;
- reusable tests;
- repeatability and consistency of tests;
- shorter release time frames;
- more confidence in the testing process;
- more productive use of resources; and
- perform difficult or impossible manual testing tasks.

Performance testing a distributed software system, such a multi-tier Web application, is a task which is much more complicated than it may look on the surface. Accomplishment of such tasks depends on how successfully one finds, deploys, and master test automation tools. The instant worldwide audience of a Web site makes its quality and reliability crucial factors in its success. Correspondingly, the nature of the Web and Web sites pose unique software testing challenges. Webmasters, Web application developers, and Web site quality assurance managers need tools and methods that meet their specific needs. Mechanised testing via special purpose Web testing software offers the potential to meet these challenges, (Mosley, 1999).

In these tough economic times, software development managers are pushing to get more and better testing done faster, (Poston and Sexton, 1992). Most of these

managers recognise that automated testing tools facilitate higher quality and more productive testing, but acquiring such tools is often quite complicated. Kavi (1996), attested to this when he stated that software quality could be greatly improved by selecting a correct tool to assist in each phase of the development process, from requirements analysis to final testing and integration. According to Mosley (1999), network and Internet software applications must be 'load tested' and monitored in terms of the volume of traffic, and this kind of testing is only doable using automated testing tools. Selecting an inappropriate tool, on the other hand, can actually hinder software development. That is why when a tool had to be chosen for use in the test experiments conducted in this study, a comparison criteria table was developed and used to evaluate and choose the most suitable tool.

Software engineering tools are more than just instruments for enhancing productivity during software development. They have proved invaluable to research and education as well. In fact, some research projects begin only after researchers are assured of getting a certain tool, (Horgan and Mathur, 1992). The market for Web site analysis and performance measurement and testing tools is small, but growing, (Bacheldor, 1999). Automated test tools can help one in measuring an application's ability to handle heavy loads, (Systest, 2001). The added advantage with these tools is that they can be employed both in a laboratory environment as well as at a customer's site. Nevertheless, it is important to note that testing tools alone will not automate the testing process. The testers must take it upon themselves to study the tools and make good use of them. This was also the responsibility of the facilitator of the test experiments of this study.

5.5.3 Challenges of Software Testing when Using Automated Testing

There are many obstacles to developing more effective testing technologies and transferring them to industry. Some of the challenges faced with automating testing are:

- automated testing tools cannot be counted on to consistently generate test data, force the execution of certain paths, and predetermine concurrency patterns (Osterwell and Clarke, 1992);
- because software testing is sometimes viewed as a dead-end job, the most skilled testers are typically eager to "move up" and become programmers,

analysts, or system architects (Weyuker *et al*, 2000). This leaves testing without the professionals who are skilled and experienced enough to conduct quality tests, and perception about testing as a transition job;

- unrealistic expectations: due to media hype about a new technology one might expect test automation to solve all software testing problems (Test Focus, 2002);
- maintenance of automated tests: this requires resources, time and effort (Test Focus, 2002); and
- false of security: No matter how many times the tests are run, they will not find the defects external to their coverage, (Test Focus, 2002).

5.6 Testing Tools Considered in this Study

The goal of this section is to determine the best test automation tool, or combination of tools, for the performance testing experiments. According to Sabin (1998), the best approach is to select the test automation tool most likely to be most compatible with the client-server product and operational scenarios selected in the test approach. The testing plans considered appropriate for this study led to the consideration of tools to use for testing the applications.

In order to choose and use a tool suitable for the test experiments to be conducted for this study, three different Web site automated performance testing tools were described and evaluated by comparing them using a number of criteria. Eisermann (2002) wrote that if someone tests the speed of a Web or a proxy server, he would not know (if his results are good or bad) if he does not make comparisons with other products. The description of each tool is mainly based on descriptions provided in the Web sites where these tools are listed, where the vendors describe what each tool is and what its capabilities and features are. There are many testing tools on the market that are designed to improve testing, but only the following could be considered for the purpose of this study because of their closeness to what this study purports to achieve. These tools are described in more details in Appendix B.

- TeamQuest Web Performance Agent
- TestMaker– PushToTest
- WebPerformanceTrainer

These tools were evaluated mainly on their capabilities, platforms, servers and client support, functionality and ease of use. Appendix B (Description of the Automated Test Tools) contains the descriptions and features of these tools. Table C.1 (Comparison Criteria) contains the comparison criteria devised in order to determine the right and most suitable testing tool for this study's needs. According to Kavi (1996), one should develop a simple, usable, and yet comprehensive criteria for the evaluation of software tools. These requirements were guiding the work in defining the comparison criteria in Appendix C.

As discussed above, the tools chosen for comparison are automated testing tools. The comparison criteria in table C.1 (see Appendix C) are arranged in general groups, environment-dependent, tool dependency, and tool non-dependency. The following section explores issues related to the evaluation of the tools listed above, which is presented in more detail in Appendix B.

5.6.1 Evaluation of the Automated Performance Testing Tools

The nature of the Internet and Web environment is open and many different products today are designed to run on different platforms. This creates additional burden to both design and testing of applications. The result is that testing on multiple platforms has become a necessity for most products. It is therefore important that tools used to test the applications support multi-platform testing, (Chillerege, 1999). Considering all the criteria used to evaluate these disparate tools' features, it seems evident that a single product that supports all of the criteria will not be possible. However, there is one common theme and that is that the majority of the evaluation seems to be based on what is expected of the tester to be able to use the tool with ease, and how much of a score is the tool getting for economic proposition, (Marick, 2002) which are examined above. The chosen tool should therefore address all aspects of ease of use, economic propositions, and multi-platform development and testing. In addition to the criteria set in table C.1 (see Appendix C) this study also employed the advice of

Grove Consultants (2001), which states how to choose a testing tool. Their recommendation is as follows:

- Start by looking at the current situation.
  - identify the problems.
  - explore alternative solutions.
  - realistic expectations from tool solutions.
  - is one ready for tools?
- Make.
  - what are the current and future manual testing costs?
  - what are initial and future automated testing cost?
- Identify.
  - identify constraints (economic, environmental, commercial, quality and political).
  - classify tool features into mandatory and desirable.
  - evaluate features.
  - plan and schedule.
  - make the decision.
  - implement the selected tool

The next section will look into the selection of the suitable automated testing tool.

5.6.2 Selection of the Automated Performance Testing Tool to be Used

Following Sabin (1998), where he wrote that selecting the best test automation tool(s) makes the testing project easier to accomplish, and delivers more success; this section examines the election of the automated tool used in the test experiments. Table C.1 in Appendix C shows a number of criteria used to evaluate the test tools considered in this project. The main purpose of the criteria is to provide a means to evaluate and compare the three automated testing tools considered for evaluation. This section discusses the results of the assessment or evaluation of the three different tools and extracts the critical and essential characteristics according to the user's requirements, and completes a tailored summary of what the score really represents.

(a) Scoring System

Since the evaluation process involves multiple criteria and many alternatives, an approach from the field of Multiple Criteria Decision Analysis called the Simple Multi Attribute Rating Technique (SMART) was chosen (Lootsma, 1999). It uses separate weighting of the criteria and the alternatives along all the criteria. Then the weights of the criteria are multiplied by the scores of the alternatives and the results are summed up. On that basis it was decided to apply a 1 – 10 scoring scale for the criteria. The most important criterion gets the greatest weight and the least important criterion gets the lowest weight. The scoring process generally involved identifying how well the tool performs. According to Poston and Sexton (1992) for such rating and scoring one must use vendor feedback and published evaluations at hand.

(b) Results of the Comparisons of the Test Tools

Appendix C and table C.1 illustrates the scores awarded to each one of the tools. Looking at the total scores, WebPerformance scored the highest, with TeamQuest second and TestMaker third. The comparison criteria table, together with the brief descriptions of the tools above, highlights the tools' good and bad points. One should note that the good and bad of these tools were primarily focused on the needs of this project. Based on table B.1 the tool that had to be used is WebPerformance. As a true enterprise performance management package, WebPerformanceTrainer Software offers more than just Web server data. It offers a look at the complete picture, including the following:

- allows one to identify cycles in system behaviour;
- drills down capabilities to quickly pin point bottlenecks (for example, one can drill down from total connections to the files most accessed);
- maintain a historical record of Web Performance values to help determine trends;
- alerting capabilities so one is aware of potential problems before they occur;
- drills across capabilities to investigate causes across applications;
- determine the root cause of problems with correlation analysis – even across applications on the same server; and
- plan for the future with trending and dynamic modelling capabilities.

Web Performance Trainer fulfilled the conditions better: graphical user interface, proxy configuration, easily understandable, and Internet based documentation/user manual. The documentation, which can be found on the Internet, is comprehensive, clearly structured and useful. The handling of the product is intuitive for experienced Windows users. There are versions for both Linux and Windows operating systems.

5.7 Discussions on the Practical Implications of Test Design Decisions Taken Based on the Theoretical Foundations

5.7.1 Action Plan for the Experiments and Testing Process

This chapter started off with an examination of a number of theoretical foundations of testing plans and methodologies. This section specifies the actions which were taken based on the theoretical foundations. The action plan for this research is based on the theoretical foundations examined above, in section 5.2. In this section, the actual plan of action is laid out.

The testing process followed was as examined above in section 5.2. Any deviation from that testing process were properly documented and reasons thereof given accordingly.

5.7.2 The Testing Plan Employed in this Study

The test plan employed in this study was the one which was suggested by Nguyen (2000), integrated with the one by Empirix. The performance testing modelled user activity during Web site usage. User activity logs from Web site usage gave some idea of the activity level for the tests which were conducted.

The *test plan design style* adopted has the following advantages:

- unless testing adjustments are required, the test will always be run the same way. One is more likely to be able to reproduce the failure or success, provided all elements remain the same;
- it details all the important expected results;
- there is knowledge of what to test; and
- it spells out all activities and tasks of the testing project.

In terms of the execution of the tests, this study will follow Musa (1996), according to whom "test execution involves identifying failures, determining when they occurred, and establishing the severity of their impact." In order to identify failures this study, firstly, looks for deviations of program behaviour, and then determine which ones affect the user. It will be assumed that only those programs that negatively affect the users are failures. Manual inspection of test results will be done to identify failures not amenable to automatic detection and to sort out deviations that are true failures. The observation of the test and documentation will focus on the items listed below. These items (listed below) will also serve as a checklist in chapter six, when the tests are documented. The checklist below is partly adopted from Hampton (2002):

- distinct output is generated for each input;
- system state can be queried during execution;
- past system states are visible (transaction logs are available);
- all factors affecting the output are visible and logged;
- incorrect performance is easily identified;
- internal errors are automatically detected through the tests;
- internal errors are automatically reported;
- changes to software are infrequent;
- changes to software are controlled;
- changes to software do not invalidate existing tests; and
- the software recovers well from failures.

5.7.3 Testing Methodology to be Applied in this Study

As part of the planning process, a testing methodology was proposed. This section looks into the proposed testing methodology which was then applied in this study. In chapter six the methodology was applied in conducting the test experiments.

After exploring a number of the testing plan designs (in section 5.2) one could now outline a basic experiment format that provides a high level of realistic loading of candidate Web site using client-side browser based testing technology. The software engineering literature contains many studies of efficacy of fault finding techniques. Few of these, however, consider what happens when several different techniques are used together, (Littlewood *et al*, 2000). It is for this reason that for the purposes of

this study only some of those items in some of the methodologies, plans, strategies, procedures and steps were employed in planning and conducting the tests. The reason for doing this was because of the limited applicability to Web environment of some of these guidelines, and thus the motivation to pick only that plan which could be useful. The questions asked in choosing the methodology to adopt is whether it is useful and whether the technique is usable. Firstly, this study adopted the methodology as suggested by Arthur (1988), where he made use of both black-box testing and white-box testing. He termed it 'gray-box', which combines the strategies/approaches of both black and white box testing. Black-box and white-box testing were discussed in greater detail in chapter two.

Secondly, for load-testing, the approach by Alberto and Weyuker (1998) was adopted. Alberto and Weyuker presented a load testing algorithm that relies on the software's operational profile to select a meaningful and manageable set of test cases. This strategy emphasises those software states which are most likely to be entered at least once.

Thirdly, as a subset of the applied methodology, the following is a summary of activities that were also implemented:

- identify user performance related requirements, including the anticipated number of concurrent users, transaction response time, and details of the operating environment;
- develop transaction mixtures, transaction work loads, data needed for the test, resources, schedule, and test procedures;
- choose a suitable tool (already done above);
- implement the test, including execution and recording of the test activity and results (which is the subject of the next chapter);
- make performance related testing adjustments (where and if necessary); and
- analyse and conclude on the testing results (which is the subject of the next chapters).

In addition to the above list of activities, the following will be also used as part of the suggested methodology:

- **Phase 1: Create Typical Tests Cases.** Using existing records, collect data of the Web sites that represent typical usage of the candidate Web site. The recordings of new sets of data was made with real time recording switched on so that all "wait times" could be kept in the data set.

- **Phase 2: Identify the Testbed.** Run a playback and emulate user behaviour with very great precision. Tests to be run without any data in the browser's caches to ensure that the tests do, in fact, really download all of the relevant URLs. According to Swanson (2004), client/server testing can be looked at in terms of testing which occurs largely on the client-side, or that which occurs largely on the server. In this study, the automated testing tool was installed on a client computer, and focused on the performance of the applications as observed from the client side.

- **Phase 3: Establish Scenarios of Tests.** The collection of tests recorded in Phase 1 needs to be organised into groups, with specific delay multipliers and playback repetition counts of each test taken into consideration. The guidelines for the scenario design includes, amongst other things, the following:
  - *Total running time.* The total number of test and repletion counts will be enough so that a "steady state" can be reached and sustained for a long-enough period of time to have confidence that the Web site performance at that level is adequate.
  - *Scale up.* The test will begin with a smaller test scenario and then scale up on the ratio of one (1), then two (2), through to ten (10) (1/2/5/10 ratio). This ratio would be increased based on the results of the initial tests.
  - *Date rate.* The total rate for all of the activity generated while the scenario runs will be kept constant.

The test will start with twenty virtual users. Each minute, 20 additional virtual users will be added at random intervals, until the test is complete. Real users will also be added. Data will be collected, summarised and documented.

- **Phase 4: Establish Load Target.** Run the test to completion, unless interrupted for some specific reasons.

- **Phase 5: Run Test.** Because the planning described is well done, running the tests is going to be straightforward. The basic load and stress experiment structure to be imposed by each scenario is known and the sequence in which to run them is explicit in the scenario definition section. Firstly, the base performance time will be established and run for a very lightly loaded server. Secondly, after this is completed, the tests will be run up in stages to target maximum load and abnormal situations. The average response time will be measured in each successive stage.

- **Phase 6: Document, Analyse and Conclude.** When the tests are completed, the test results will be documented. The test results will then get analysed, and conclusions drawn there from.

5.7.3.1 Tool to be used in conducting the performance tests

In section 5.2, automated performance testing tools were evaluated using multiple comparison criteria. A tool was singled out as the one considered appropriate to use for this research. This tool is WebPerformanceTrainer. It will be used to test the performance of the Web site. The test data will serve as an input into the tool.

5.7.3.2 Special Cases Which Are Likely to be Considered in This Study

During the test execution special cases will be considered. These can be listed as:

- cascaded failures
- repeated failures
- failures that will deliberately not be resolved

Depending on how material and significant these are, they might or might not, affect the final results of the tests. Of these special cases only those that are material and worthy of reporting will be noted in the test documentation.

5.7.3.3 What Tests Will Be Included In The Experiments

The goal in this chapter was to come up with a comprehensive approach for planning and executing the performance testing experiments. The approach adopted in this study hinges upon the use of a Web site to verify that performance falls within normal usage bounds, then stress test the Web site by increasing the load over the normal levels. It is not planned that this approach will lead to critical performance improvements which will be dealt with in this study. As it was stated in chapter one, this study will only focus on conducting the performance tests, documentation of the observations, and the analysis of the test results. Any performance issues, if required after completion of tests, will be reported to the Web site owners, and it will be their duty to decide on the action(s) to take to resolve such issues. In section 5.1 above the role of the facilitator of the experiments was discussed. Conducting and the supervision of the experiments was the task of the testing facilitator.

5.7.4 Other Issues Related to the Testing Plan

5.7.4.1 Choosing Sections to be Tested on the Web Site

The popular pages for use during the test was selected by using the log file for checking those pages whose every request the server accepts, including information such as date and time of the request submitted by the clients. According to Chevalier and Quarterman (2002), focusing on cached Web site speed says nothing about routers, dynamic routes, or worldwide performance of all networked computers. Care was taken not to test for performance with cached Web pages, no matter how frequent they are used. This required clearing of the cache memory before the tests were started.

The Web site was tested with the same version of the test data and test cases. This ensured that the output from all the tests was better analysed and concluded upon. It only depended upon material and special cases where a different version would be used.

5.7.4.2 Sections to be Tested

It is understood that the 'look and feel' of a Web site has an effect in terms of attracting and keeping Internet surfers. Since it is not within the scope of this study to investigate such factors, this was overlooked. Nevertheless, to ensure dependability

and performance during the tests, the functionality and usability of the Web site was briefly tested. The Web site's functionality was tested because functionality testing has more to do with testing the site's application and whether its components are linked and working together and meeting its stated requirements. The issue of testing for functionality and usability has already been discussed, and its necessity for this research was established.

5.7.4.3 Assumptions

The following are the assumptions which were made:

- it is assumed that the system and its infrastructure will remain responsive and stay up and running during the tests, unless brought down by the increasing loads of virtual users during the tests; and
- the version of the subject Web site remained the same throughout the testing effort.

5.7.4.4 Test Context and Environment

The tests operated from the browser level for two reasons; firstly, this is where users see a Web site, so the tests were based on browser operation because this is more realistic. Secondly, the tests were based on browsers because they could be run locally or across the Web equally well. Local execution is fine for quality control, but not for performance measurement work, where response time including Web variable delays reflective of real world usage is essential.

The tests were conducted on environments which the researcher could not change. This means that the setup of the testing environment was accepted as was. In a way, this can be considered as an advantage to the testing activity because the test results were yielded from real life environments.

5.7.4.5 Test Mode

The modes of tests were examined in detail in chapter two. It was stated that one may run tests in a variety of modes, and this is what was done when conducting the tests. The modes that one could plan to use are background testing, distributed testing, performance testing and random testing. The scope of this study limits the activities of this research to the performance testing mode.

5.7.4.6 Deliverables of the Tests

The deliverables associated with this testing project can be listed as:

- the testing plan, which is already examined in the sections above
- test cases;
- documented test results and observation (in the form of chapter six);
- analysis of the test, which will be carried out in chapter six; and
- conclusions on the test results and analysis, which will be in the form of chapter seven.

5.7.4.7 Deviations from the Testing Plan/Methodology

Marick (2002), warned against sticking stubbornly to the testing plan as one of the testing mistakes to be avoided. He wrote that good testers are systematic and organised, yet they are exposed to all the chaos and twists and turns and changes of plan typical of a software development project. This means that there is a possibility that the tester might deviate a little from the testing plans, depending on whether there is a need for this or not. A perfect example could be the use of the same version of test data and test cases for the Web site.

5.7.4.8 Post-Test-Session Questionnaire

It has already been established that there is a need to get feedback from the real users. The feedback will support the decisions made in terms of how the Web site performed. It was decided that a questionnaire would be used to get user feedback about the performance of the Web site during the test experiments.

Even though it was not the primary methodology employed in this study, the questionnaire in appendix D was used to gather feedback from the real users. Bouma (2000) wrote that it is often better to use several data-gathering techniques to answer a research question. The goal of using a variety of techniques may provide different perspectives on the situation at hand, thereby increasing what is known about it. The questionnaire is the generic set of questions and a tailored set of functional questions specific to the subject Web site. It is understood that for the questionnaire respondents to be considered for statistical purposes they should range from thirty to about hundred, but due to the fact that the questionnaire is not the main tool used in this research, the number of returned questionnaires is considered immaterial. Also, the

number of participants will not be material because the questionnaire was used in this project as a supporting material and its findings were proclaimed as exploratory and not definitive. The following discussion is on the variables (both dependent and independent) and the propositions of the test experiments for this research.

5.8 Propositions, Dependent and Independent Variables Considered in this Study

A number of variables are considered for use with the propositions defined below. The propositions are stated in table 5.2 below. The requirement to satisfy online shoppers is usually translated to making sure that one's Web site performs according to users' expectations. This can be achieved by testing the Web site to discover the amount of load and stress the system can handle at a given time. To run such tests this study experimented on the changes on the performance due to increased and decreased usage. For the experiments to be successful, dependent variables as well as the independent variables were defined.

5.8.1 Dependent Variables

The test experiments conducted determined how the dependent variables were affected. The identified dependent variables were (at a certain point in time), the *response time*, *throughput*, and *data access rate*. All these variables are components of what could be termed *performance rate.*

5.8.2 Independent Variables

The identified independent variable is the *'number of users'*. Using *performance rate* as the (manipulated dependent) variable, the basic approach was to change (vary) the Web application's *number of users* (independent variable) and then measure the Web application's *response time*, *throughput*, and *data access rate* (the dependent variables).

5.8.3 Nuisance Variables

There was a need to take into account other factors (nuisance variables) that could also affect the dependent variables. These were the *cache* and the *network traffic*. In order to have these (nuisance) variables controlled, it was planned to have the network traffic free of other users other than those of the test experiments. The cache was cleared and cookies cleaned from the client computers to avoid Web site recognition by any of the computers, thereby affecting the performance rate.

5.8.4 Propositions to be Tested

Defined in this section is an overall proposition and a set of seven (7) propositions. It should be determined whether or not the increased concurrent use of the system has any effect on the dependent variable(s). Table 5.3 below illustrates the propositions. In addition to the seven propositions an overall proposition was defined, and it was stated as:

Overall Proposition: The extent to which the performance of a Web-based e-commerce application gets decreased is directly related to the number of users using the system at a given point in time.

Table 5.3: Propositions in the Test Experiment

Description of Propositions		
No.	**Effect**	**Description**
1	Overall performance rate not affected	There is no relationship between the decreasing of the performance rate of a Web-based application and the number of its concurrent users.
2	Overall performance rate decreases	There exists a decrease in the site's overall performance rate due to the increased concurrent users of the Web application
3	Failed requests and errors	There exists an increase in the number of failed requests and errors due to the decreased performance levels

..... Table 5.3 (Continued)

4	Number of online shoppers	There exists a drop in the number of concurrent users using the Web site due to the decreasing performance.
5	Performance in workload	There exists a change in the performance rate in the dimension of workload due to increased numbers of concurrent users.
6	Performance in stress	There exists a relationship in the performance rate in the dimension of stress due to the amount of time spent on the Internet operations, the difficulty of Internet access and line congestion, and response time.
7	Acceptable performance	There exists a change in the acceptable level of performance rate due to the levels of decreased performance.

For the purpose of this study, performance, as mentioned in almost all of the above propositions, will include response time, throughput, and data access rate.

5.8.5 Conducting the measurements

The process of taking the measurements was through the use of an automated testing tool. Once the test bed was prepared, and the test cases properly entered, the tool stored all the test figures reporting the progress of the test from start to end. The results were then exported from the test tool to a spreadsheet, where they (results) were statistically and/or mathematically analysed.

5.9 The Statistical Approach to Analysing the Results

According to Nielson (2003), the minimum acceptable number of waiting time by the user of a Web site is three (3) seconds, and the maximum is 10 seconds. Based on this premise, control charts were used for monitoring the Web site's performance. The minimum and the maximum number of seconds given above were used as the control limits. According to Bowerman, O'Connel and Hand (2001), theoretically, control limits are supposed to be computed using subgroups collected while the process is in statistical control. However, it is impossible to know whether the process is in control until one has constructed the control charts. The charts were constructed in the next chapter, when the test results were collected and documented. Below is an example (figure 5.1) of how the control chart looks like.

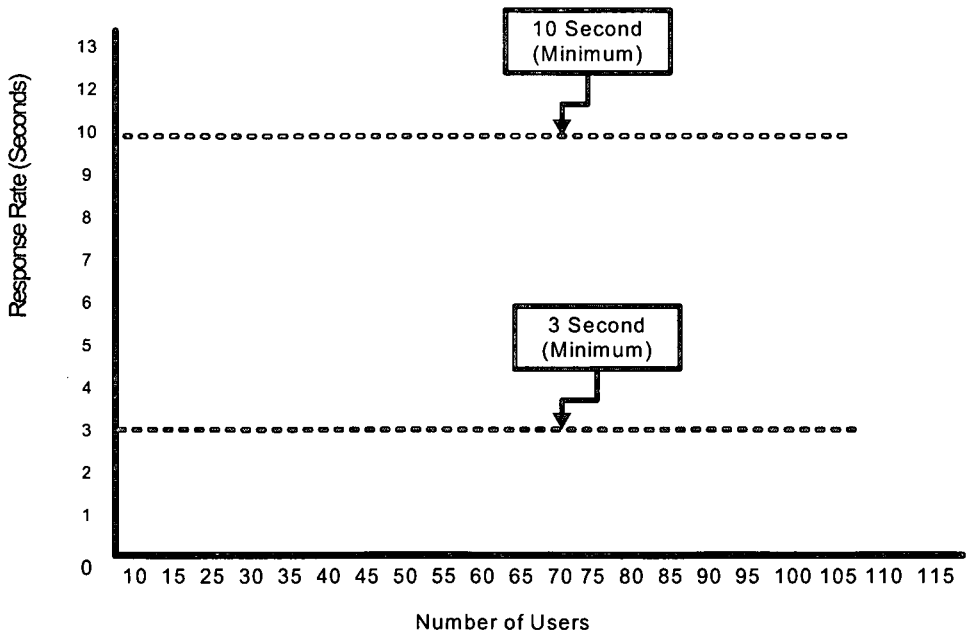

Figure 5.1 – Example of a Control Chart

The curve showing the performance rate (*response rate*, *throughput*, *data access rate*, *stress* and *work-load)*, was plotted on the graph and illustrated the performance levels where and when the minimum and the maximum control limits were exceeded. According to Gartner (2002), the maximum level of recommended performance level

is eight seconds. This was not plotted in figure 5.1. In the next chapter, the charts were constructed with the eight seconds maximum limit included.

The process of statistically analysing the results involved the usage of *confidence interval* (where the point of discontinuity is identified), and *rolling coverage* or *progressive total* (used to identify the range within which the bottleneck and discontinuity occurs), and thus state the points or range where intervention would be necessary or needed. These two statistical analysis methods were applied when the control charts were drawn up. The next chapter shall focus on this.

5.10 Concluding Remarks

This chapter presented several industry-standard-testing-methodologies. They all provide a yardstick for preparing software testing for software development processes. The testing plans were analysed and a choice made on the one to employ in this study.

A methodology for conducting the test experiments was also discussed. One needs to understand the nature of the application to be tested to be able to adopt the methodology which would better suit the software project at hand. The methodology employed in this study is based on the understanding of the nature of the Web applications which were tested.

This chapter also examined test automation and automated testing tools. This revealed that automating the software testing processes has a lot of advantages to offer, especially for performance testing, with a focus on stress and workload testing. Automated testing tools were evaluated by using a comparison criteria table (see Appendix B), which was designed specifically to evaluate the three tools under consideration, and the one suitable for this study was chosen.
The propositions, which were checked (in chapter 6) whether or not they are supported by the test results of the test experiments, were also defined. These

propositions played a very important role in analysing, discussion and concluding on the achievement of the goals and objectives of this study.

The next chapter employs the selected automated testing tool, and the methodology, together with the testing plan adopted in this chapter.

Chapter 6: Documentation of Test Experiments

6.1 Introduction

This chapter is focused on the documentation of the observations of the tests conducted on the Web site using the automated testing tools employed in this activity. According to Yamaura (1998), documented tests can be beneficial in several key ways, which include:

- *A chance to analyse the specification from a different angle.*
- *One can repeat the same test*
- *Somebody else can execute the test*
- *One can easily validate the quality of the test*

Thus, in order to precisely analyse and discuss the results of the test experiments, the tests were documented in the form of this chapter. The test experiments were comprehensively documented and analysed. The discussions on the planning of the test experiments were done in the previous chapter. Two main tasks were performed. The first task was an establishment of the number of users to scale up with, and the second one was the observation and documentation of the system's response.

6.2 Non-Disclosure of Web site Names and IP Addresses

It should be noted that information regarding the Web site, namely content, name (company name) and Web site address, is very sensitive. In order to be given permission to conduct the tests on the Web site, non-disclosure agreements, which restricted what could be explicitly documented about the site, had to be signed. Despite this fact, all testing plans, test results, etc are not restricted from documentation in this study. The restrictions were only applicable to the name of the organisation and the Web site address. Due to this restriction, the subject Web site was simply referred to as "the Web site" or just "the site". The Web site has been in

use for almost a year, and was never subjected to any form of performance testing in the past.

6.3 The Role of the Facilitator

As already discussed in the previous chapter, the test experiments had to be supervised by a facilitator. The author acted as the facilitator in all test experiments. The facilitator's role was limited to setting up the cases for a playback, as well as explaining (to the real users) the task, reinforcing the objectives of the testing sessions, explaining the basic principles of online shopping and how the e-commerce environment works. The facilitator was responsible for administering the process, time keeping, and ensuring the participants moved onto the next phase when required to do so. The facilitator also explained the questionnaire, which is discussed later in this chapter. The facilitator was not to arbitrate on indifference of the type of actions to take when certain experiences came the user's way. He merely observed the process and arbitrated on procedural issues.

6.4 Information on the Documentation of the Test Experiments

This section provides a report of the experiments conducted on the Web site under different levels of performance. According to Eisermann, (2002) the most important results, if one tests the performance of a Web site, are described with the following data: Total hits, errors, hits per second, and bytes per second. The data is presented along with these lines, with analysis and commentary. Results reported in the form of hits per second and page duration have little value by themselves as they are in raw number format. According to Merrill (2002) only when data are compared against the other (comparable) data do they provide valuable results on which to make performance judgments. Thus in this research only after reporting all the tests could the analysis be done on a comparable basis. Below is the report regarding the experiments.

6.4.1 The Testing Environment

The tests were conducted in a computer laboratory. The computer laboratory was equipped with over 35 computers. Internet connection from this laboratory was via a gateway server, which also does caching of frequently visited Web sites and Web pages. The cache was cleared before the experiments were started.

6.4.2 Test Case Entry

As discussed in chapter five, a proprietary application or tool, WebPerformance Trainer, was used to run the tests and to report the results and analysis of the tests. Before the test execution, all the test cases were uploaded to WebPerformance Trainer and the results were collected.

6.4.3 Maximum Load Definition

The maximum load definition involved running the tests on the Web site to define the maximum number of users and processes that could be handled within the recommended performance levels (minimum and maximum). The users were scaled up as the test runs were executed. These scalability tests were performed with different numbers of users at different time intervals each, and helped in determining the maximum load. The Web site owners later used these results for reliability and stability tests.

6.4.4 The Tested Web Site

Before the test experiments commenced, the test facilitator checked and tested the Web site for the following:

- the Web site does not crash;
- the Web site does not hang for more than three minutes;
- the Web site service rate is 99 percent or more;
- the client's disk usage stays stable and temporary files are deleted as expected;
- the number of processes is as expected, and there are no locked processes, and the processes do not crash;
- there are no memory leaks;
- there are no errors in outputs and log files;
- functionality and usability are satisfactory.

After having briefly tested the above, the facilitator concluded that the Web site was in a good condition and ready to commence with the test experiments.

6.5 Information Regarding Test Participants

6.5.1 Assignment for Participants in the Test Experiments

This subsection refers to the "real users", and not the virtual users, who were part of the experiment. Once the experiments were set up, the challenge was to have users access the site and behave like "typical" users of this type of e-commerce Web site. Users navigated and browsed the site, searching for information and perhaps purchase books if the material and price were appropriate for their needs and budget. An incentive to interact with the Web site and to purchase items was offered in the form of huge discounts for the participating users.

To achieve the goals of this study, the participants were instructed to follow a three step process. Firstly, they completed an online form with simple demographic data. This step was required only for the first session of each user. Secondly, the description of music categories to use as basis for the search was made available to the participants. The participants needed to select two items from the CD categories. The final step required the participants to access the online shop's Web site to select the most appropriate CDs for the type of music they selected.

To select the most appropriate music category, the participants had to search and browse until they found those music categories that they considered most appropriate. The facilitator did not provide any definition of appropriateness of the music categories, and thus the selection meant different things to different users, which is supposed to lead to a variety of activities, and the varied effects on the performance of the Web site. However, as mentioned before, an incentive was provided so that the users took the completion of the task seriously. The instructions indicated that at the end of the experiments, five users who completed the most appropriate tasks for the experiment would each receive a fifty percent discount on the CDs purchased. Further directions specified that, if more than five users completed the most appropriate tasks,

the amount spent would be evaluated, and ties would be broken by considering the time spent on the Web site. Again, the objective was to create the conditions observed for similar Web applications, and to make it relevant to this study.

A list of the participants was assembled, containing primarily students from the department of Information & Communication Technology. The list also contained a number of participants from an Internet consulting company. An email stating participation requirements and incentives was sent to the participants on the list. The e-commerce Web site was made available to the users for a period of a week before the day of the experiments. Users were restricted to the use of Internet explorer 6 as that was the preferred browser for the Web sites.

For the purpose of experimental treatment, to prepare the participants for the experimental tasks, they (participants) were given the original documentation defining the Web site and the experiments. Twenty minutes was spent explaining the testing procedure, and ten minutes explaining the test experiments

6.5.2 The Collection of the Data

The data collected was generated from a set of five different batches of experiments. Each batch of the test contained a set of five different test runs. The different batches were run at different times of the day and night. During the experiment, the data illustrating the workload, performance levels, user capacity, peak duration, and duration threshold were collected.

6.6 Information about the Subject Web site

6.6.1 Description of the Web site (Online CD shop)

The Web site was e-commerce based. It was an online CD shop, and provided the following features and functionality:

- an interface for customers, to browse the catalogue and to buy CDs;
- an interface for the Web site administrators, to administer the Web site and the inventory sold on the Web site;
- databases for the inventory and customer accounts;
- applications to manage customer accounts, shopping cart objects, and inventory;
- a subsystem to track and fill back-orders (orders to be filled for CDs that are yet to arrive).

As per the goals of this study, the experiments focused on the functionalities that were accessible to the customers. The following actions were included:

- *browse and view*: a list of items retrieved from the database, which also had the descriptions of the items;
- *add to cart*: an item could be removed from the customer's shopping cart;
- *remove from cart*: an item could be added to the customer's shopping cart;
- *checkout*: the customer could buy the items in the shopping cart;
- *login*: the customer's login would be validated and if valid the customer's account information was retrieved from the database;
- *register*: a new customer was registered with the CD store.

The administrator section was supported by mainly two functions:

- *inventory update*: the stock for each item is updated
- *backorders filling*: outstanding backorders were filled (following inventory update)

As already stated above, the administrator section was not part of the experiments because it fell outside the scope of this study.

6.6.2 The Technical Data/Details of the Web Site

Customer functionality was implemented through VBScripts and modules to handle data and the dynamic generation of HTML pages, an Oracle database to manage database accesses, a database structure composed of 11 tables for tracking CDs, transactions, and other objects, and JavaScript and cookies to provide identification and personalisation functionality.

The database was populated with information about over thousand of CDs. This information included such things as title, artist, short description, category, price, rating, etc. For reasons of making the testing more accurate, as well as the fact that virtual users were employed, the Web site was adapted to expedite the registration procedure and minimise logins. Additional technical data about the Web site is as follows:

- platform: Windows NT, Windows 2000, Sun Solaris, HP-UX, IBM AIX;
- database: DB2/UDB 7.2 and Oracle 9i;
- webserver: iPlanet 4.1 SP9, IIS 5 and IIS 4;
- application server: WebSphere 4.0.1 and IBM JDK 1.3, Weblogic 5.1 SP2 and JDK 1.3, BEA Weblogic 6.1 SP2+ JDK 1.3;
- tool used: WebPerformance Trainer for automated test execution and analysis of test results.

6.7 The Results of Test Experiments and Facilitator Observations

Conducting the experiments involved executing performance tests on the Web site for the purpose of achieving the goals of this study. The tests were executed at different time intervals and in a form of a batch. Tests were repeated where and when the need arose at different intervals for investigative purposes. Such cases were brought about by situations where the test results were suspicious to the facilitator. Rerunning the tests meant that consistency was checked. This means that if the results from the test

rerun were the same with the suspicious test results, then it would be accepted that there is consistency and thus the results would then be accepted.

During the test planning, which was the subject of the previous chapter (five), propositions were defined, and variables (dependent, independent and nuisance) were identified. The independent variable was the *"number of users"*, the dependent variables were *response time*, *throughput*, and *data access rate*, which were collectively called *performance rate.* The different tests were run with different numbers of users, and the observation was that as the number of users was scaled up, the performance rate would deteriorate. The inspection for support of the propositions defined in chapter five will be discussed in the later sections of this chapter.
The nuisance variables, which were kept constant during the experiments, were the *cache* and the *network traffic.* In the case of the network traffic, the network was kept free of users other than those who were part of the test experiments. The cache was handled by making sure that it was cleared, and cookies cleaned.
The Web site had a total of fifty-three functions. To be able to test the performance of the site and cover most, if not all, of these functions, a total of three test executions were required of each test suit. After the test suits were generated, tests were run and observations and documented made.

There were multiple active customers, with the rest being virtual customers or users. The workload was gradually increased without giving the current users a chance to think through and finish their already initiated tasks. After running the tests, Web Performance Trainer analysis reports provided quick answers to the key questions about the performance of Web site, such as "what was the longest load time for the Web site?" and "how many users could the Web site handle?" Below is figure 6.1, illustrating the playback of one of the experiments setup.

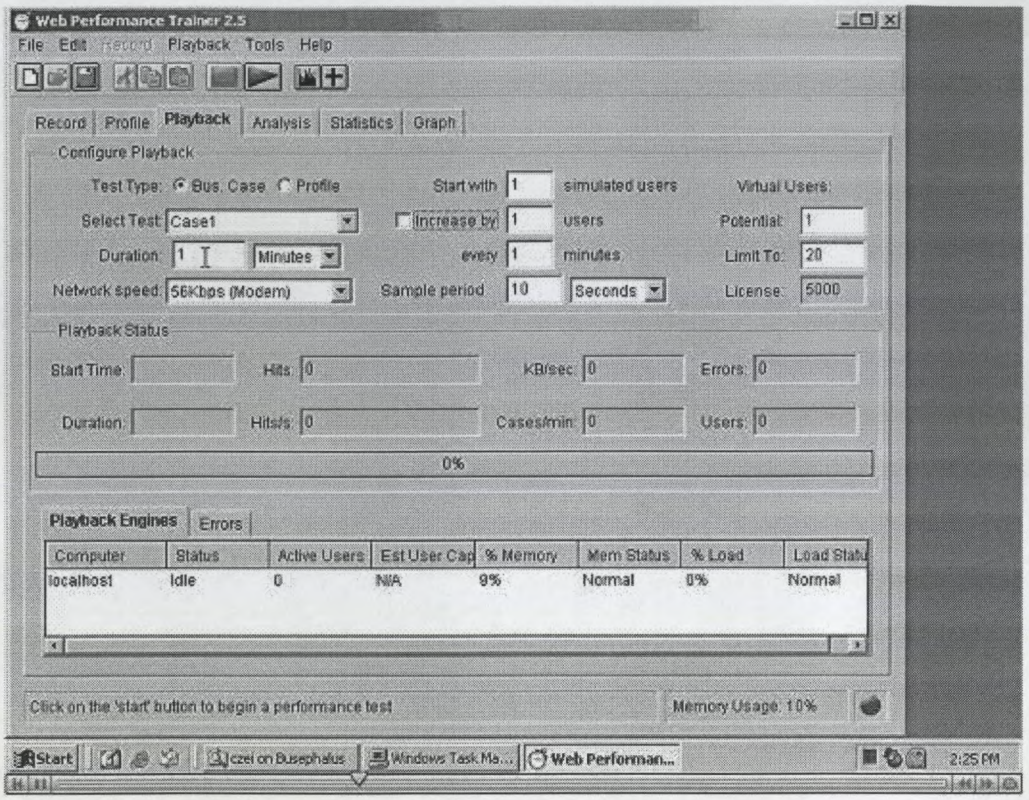

Figure 6.1: Playback of the test run on Web Performance Trainer for Experiment

The test runs were commenced with 5 real users. In addition to that, the above figure illustrates that the test run was commenced with 1 virtual user. Each test was set to run for between ten and twenty minutes. When test configurations were completed the tests were let to start. There were multiple active customers, with each customer operating at a different function and level at different times. The results of executing the test suite generated by experimental test 1 are presented in table 6.1.

Table 6.1: Summary Data Batch 1 of the Test Experiments

No.	Time	Repeats	MinTFB	Avg TTFB	Max TTFB	Min Dur	Avg Dur	Max Dur	Users	Errors
1	2:24:00	39	0.00	0.00	0.00	3.43	3.44	3.44	20	0
2	0:20:20	45	0.00	0.00	0.00	3.44	3.44	3.44	20	0
3	0:20:30	46	0.00	0.00	0.00	3.44	3.44	3.44	22	0
4	0:20:40	46	0.00	0.00	0.00	0.00	0.00	0.00	24	0
5	0:20:50	48	0.00	0.00	0.00	3.44	3.44	3.46	24	0
6	0:21:00	50	0.00	0.00	0.00	3.42	3.42	3.42	27	0
7	0:21:10	52	0.00	0.00	0.00	3.44	3.44	3.46	27	0

The above table displays tests results which were collected from the first batch of the test experiments. A total of five batches, each containing seven test runs of the test experiments, were collected. The other batches can be viewed in Appendix E. The meanings of the abbreviations of the columns in the above table (6.1) are as follows:

- No. = Number of the test. 1 means the first test (of batch one);
- Time = This column contains the time the values in the row were sampled;
- Repeats = The cumulative number of times the business case, Web pages, or URLs were repeated. Each virtual user is assigned a business case to execute over and over and this number tells how many times each business case, Web page, and URL was repeated;
- MinTFB = The minimum time to first byte during the period
- AvgTTFB = The average time to first byte, or the time it took for the Web server to first respond to the HTTP requests;
- MaxTTFB = The longest time the Web server took to respond to the HTTP request during the time period;
- Min Dur = The minimum time it took for virtual users to receive all of the business case, Web page, or URL back from the Web server;
- Avg Dur = The average duration it took for the virtual user to receive all of the business case, Web page, or URL back from the Web server.
- Max Dur = The maximum time it took for the virtual user to receive all of the business case, Web page, or URL back from the Web server. This is the slowest a user would have viewed the results;
- Users = the total number of users running at the end of the sample period.
- Errors = The number of errors that were generated during the sample period by either the selected business case, Web page, or URL.

These results were collected using the test tool. The table shows that a total of seven tests were run for this batch of tests. In fact, all the batches of the tests contain a total of seven test runs. The results in the table show a user capacity which is not as high as defined in the technical specification of the Web site. Only a total of twenty-seven (27) users within the time frames allowed for this batch of tests could be able to use the Web site.

Table 6.2: Summary Output of the Statistical Analysis of the Test Results: Dependent Variable is Average Duration

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.93386536
R Square	0.87210451
Adjusted R Square	0.868228889
Standard Error	1.542079006
Observations	35

ANOVA

	df	*F*	*Significance F*	
Regression	1	225.0231718	0.0000	
Residual	33			
Total	34			
	Coefficients	*Standard Error*	*t Stat*	*P-value*
Intercept	1.547569669	0.50643406	3.055816724	0.0044
Users	0.027882631	0.001858746	15.00077237	0.0000

Some of the statistical terms in the above table are brief defined below. Where necessary, the brief definitions are stated with their relevance to this study, and the formula used for the calculation.

In table 6.2 above the *observations* is equal to thirty-five (35). This table shows the total number of the test experiments which were conducted. As already stated above, a total of five batches, with each batch having a total of seven tests runs, were conducted for this study. The other statistical terms in table 6.2 are defined as follows:

- ***Multiple R:*** This is the Multiple Correlation Coefficient; it shows the strength of the relationship between the dependent variable (average duration in the case above) and the independent variables (number of users in the table above).

- ***R Squared***: This is the square of the correlation coefficient – and R-squared is the percentage variation in the dependent variable which was due to the variation in the independent variable. As shown in table 6.2, it could be said that 87% of the variation in the average duration was due to the variation in the number of users on the system.

- ***Adjusted R Square***: This is similar to r-square, but it is more accurate since it takes into account the number of observations as well as the number of independent variables in the model.

- ***Standard error:*** This is the "standard deviation" of the regression model's distribution.

- ***df:*** Degrees of freedom. This value is directly related to the number of users. The F-distribution values depend on the degrees of freedom in the model.

- ***F-value:*** This is the value on the F-distribution that is used to determine the probability (significant F). The regression model follows an F distribution.

- ***Significance F***: This is actually the p-value of the regression model. In this case the regression model is highly significant. The next point has more information regarding the 'significance'.

- ***Intercept***: The regression model is an equation which includes an intercept. The fact that the p-value is less than 0.05 means that the intercept is significantly different from zero.

- ***Coefficients***: These are the beta-values of the different terms in the model. The model can be written as:
  - Average Duration = 1.5476 + 0.0279 * Users + random error.
  - i.e since there are 409 users then =>
  - Average duration = 1.5476 + 0.0279 * 409
    = 12.9587 + random error

- ***t stat***: The coefficients in the model follows a t-distribution. This t-value is used to determine the significance of the coefficients in the model.

- ***P value***: This value shows that the coefficients in the model are significantly different from zero.

As part of sub-goal seven of this study, there was also a need to investigate whether the subject Web site could meet the recommended performance levels for an average performing Web site. In order to accomplish this sub-goal the graph (figure 6.2) below was constructed.

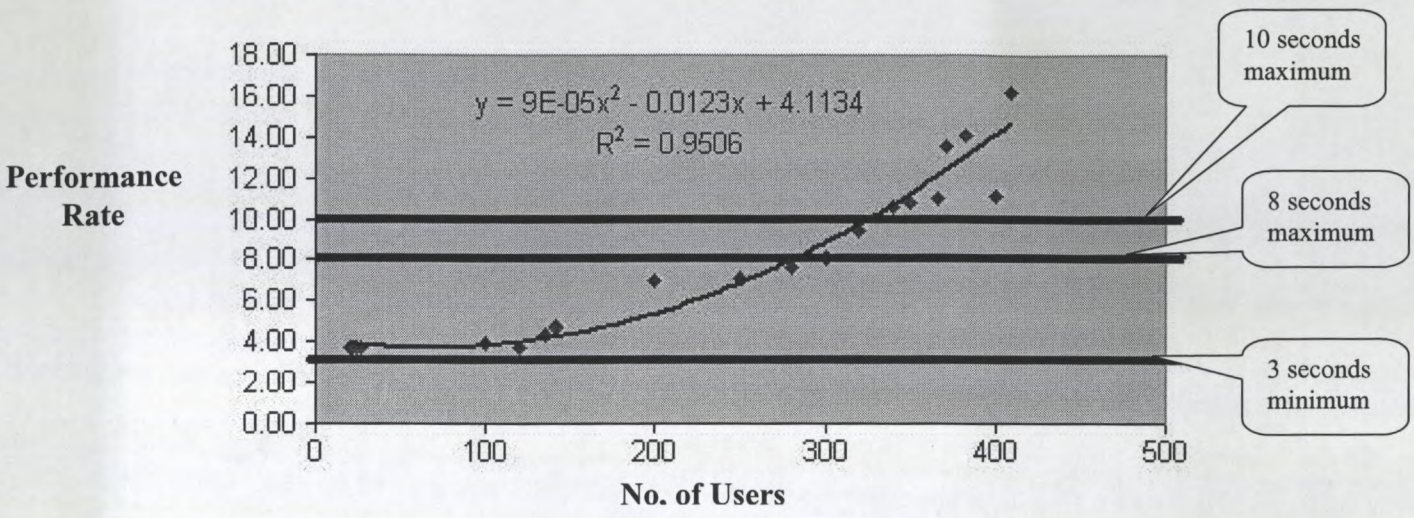

Figure 6.2: Results on the performance rate as a function of the number of users approximated with a polynomial regression equation of order 2

The above graph illustrates a control chart, which depicts the performance results of the test experiments. It is based on the results of all the batches of the test experiments conducted for this study. The batches with the test results of the test experiments are documented in Appendix E. The nature of control charts was discussed in the previous chapter. For more information see Bowerman, O'Connel and Hand (2001). The horizontal solid lines on the graph were placed later to illustrate the minimum and the maximum limits of the recommended and acceptable levels of Web application performance, as suggested by Nielsen (2002), and Gartner (2002). There are a total of three horizontal solid-lines on the graph. The one at the bottom shows the minimum acceptable level of performance, and the above two show the maximum levels of performance. According to Nielson (2002) the maximum level of acceptable performance is 10 seconds, and thus the "cloud" pointing to the line stating the 10 seconds maximum. The other horizontal solid line with the eight seconds maximum "cloud" shows the maximum acceptable level of performance which was suggested by Gartner (2002). The chart is discussed further in the section below.

As part of the sub-goals of this study, the point at which intervention would be needed to handle the performance had to be investigated. The above figure (6.2) was not good enough to pinpoint such a point. A better way to do this is by quoting a range of values. The reason for this is because one point which is out of range is not good enough to warrant a call for intervention. A point of intervention would be that where and when the system's performance stays out of unacceptable levels for a continued period of time. A statistical way of finding that is by using a running or rolling total of the average duration performance rate. The chart (figure 6.3) below was constructed for such purposes.

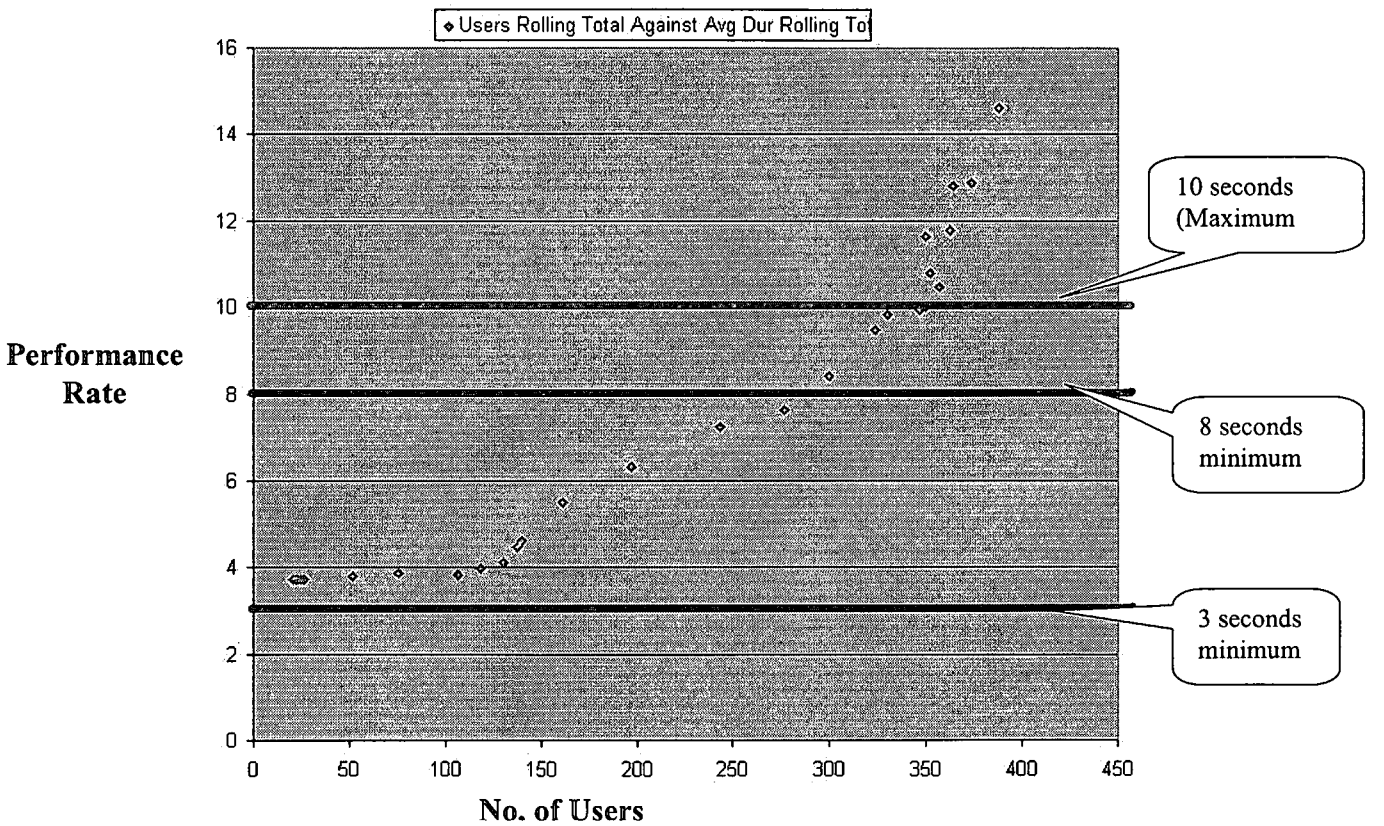

Figure 6.3: Results Output - Graph with Rolling Average Duration of Performance Rates

As briefly discussed above, figure 6.3 illustrates the charting of the rolling totals of the average duration of the performance rates of all the test experiments, which was constructed for the purpose of investigating the point where intervention would be necessary. The meaning of the "second's clouds" in the graph are same as those discussed above in figure 6.2. The point where interventions could be necessary is when the performance has is continuously above the quality performance levels

(Bowerman *et al*, 2001). According to Bowerman *et al* (2001) three types of quality can be considered: *quality of design, quality of conformance,* and *quality of performance.* Quality of performance is how the product or service actually performs in the market place. Thus the investigation of the quality of performance of the Web site in order to discover the levels where it is continuously above the quality performance levels, and therefore decided whether or not to intervene. A situation which would call for intervention is that which has the performance level staying continuously above the maximum acceptable level of performance (Bowerman *et al,* 2001). If interpreted according to Nielson's (2002) suggestions, the maximum acceptable performance level is 10 seconds. In the above graph (figure 6.3) the point where intervention would be necessary is when the total number of users is around three hundred and fifty (350). That was the point (range) where the performance level was stabilised at 10 seconds, and the number of users was about 350. If one considers the suggested maximum acceptable level of performance according to Gartner (8 seconds), then the number of users is approximately 290. The intervention would take actions to improve on the performance, including properly informing the user of the causes of the delays, and thereby avoiding frustrations on the part of users who might instantly abandon the site as a result of unexplained poor performance. The methods and ways of how the intervention would be implemented fall outside the scope of this study, and thus would not be discussed here. One of the sub-goals of this study was to only investigate and report on such a point. When this was reported, the owners of the subject Web site went further to investigate the methods of interventions. More work was done in order to statistically interpret and analyse the results of the test experiments. Below is the presentation of this work.

6.8 The Analysis and Discussions of the Results

When a linear regression model, having the number of users as the only independent variable, was fitted to the data, the r-square and the adjusted r-square showed that approximately 87% of the variation in average duration of the performance level could be explained by the variation in the number of users. However when the number of users was plotted against average duration in a scatter plot, it seemed that

the relationship between the two variables is a curve-linear relationship rather than a linear relationship, which is illustrated by figure 6.2. When a polynomial regression curve was fitted to the data the r-squared value (explained variance) increased to 95%. These results show a poorly performing Web site. This was indicated by the fact that the Web site did not even meet the minimum performance level of three seconds. In figures 6.2 and 6.3 above, the independent variable (users) was initially at 20, and that produced an average duration of 3.7 seconds as the performance rate.

An analysis from the server side revealed that a heavily loaded and poorly performing database server was causing the bottlenecks and thus negatively affecting the performance level. The database was not designed for Web use, and thus the reasons for bottlenecks at the database server level. Further analysis of the server side was not carried out as it fell outside the scope of this study. Nevertheless, the owners of the Web site conducted a thorough analysis of the Server side as it was more suspect to the poor performance results. As per figure 6.2 and figure 6.3, the results revealed that the Web site failed to meet even the minimum required level of a Web site's performance, which is 3 seconds; which is what users expect as the minimum or quick performing Web site. Figure 6.2 shows that the performance level starts at 3.7 seconds, which is above the minimum quality control level of 3 seconds.

- ***Discussions on the Propositions Defined in Chapter Five***

The propositions which are examined here were defined in the previous chapter, but are only revisited here for discussion purposes and checking whether or not they are supported. The first proposition made was:

Proposition One (P_1): There is no relationship between the decreasing of performance of a Web-based application and the number of its concurrent users.

When examining the graph of the regression in figure 6.2 and the rolling totals in figure 6.3, it is evident that this proposition is not supported. There is evidence that the performance rate decreases as the number of users is increased. This can also be seen in the statistical results of Table 6.2. For example, when the performance rate was 3.733, the number of users was 24. When the number of users was increased to

200 then performance rate decreased to 7.0167 average duration (in seconds). As a result of these observations, the first proposition had to be rejected, and thus the turning of the focus to the other propositions. The overall proposition is stated below:

Overall Proposition: The extent to which the performance of a Web-based e-commerce application is decreased is directly related to the number of users using the system at a given point in time.

The other propositions (P_2 - P_7) will be discussed before the above overall proposition. This will make it possible and easier to state whether or not the overall proposition is supported. Attention is now turned to the other propositions, which are re-listed in the table below for reference purposes.

Table 6.3: Proposition in the Test Experiments

No.	Effect	Description
1	Overall performance rate not affected	There is no relationship between the decreasing of performance of a Web-based application and the number of its concurrent users.
2	Overall performance rate decreases	There exists a decrease in the site's overall response time due to the increased concurrent users of the Web application
3	Failed requests	There exists an increase in the number of failed requests due to the decreased performance levels
4	Number of online shoppers	There exists a drop in the number of concurrent users using the Web site due to the decreasing performance.
5	Performance in load	There exists a change in the performance rate in the dimension of workload due to increased numbers of concurrent users.

..... *Table 6.3 (Continued)*

6	Performance in stress	There exists a relationship in the performance rate in the dimension of stress due to the amount of time spent on the Internet operations, the difficulty of Internet access and line congestion, and response time.
7	Acceptable performance	There exists a change in the acceptable level of performance rate due to the levels of decreased performance.

In general, the results in figure 6.2 and figure 6.3 supports the propositions. The overall performance rate decreased as the number of concurrent users increased. More about the propositions is discussed below. P_1 has already been discussed above. Therefore, following discussion will start with P_2.

- ***P_2:*** This proposition was supported by the fact that (when studying both figure 6.2. and 6.3) there is an observation of a decrease of the performance rate as the number of users is increased. At 24 concurrent users of the Web site, the performance rate was sitting at an average duration of 3.7 seconds. That is slightly above the suggested or recommended minimum level of 3 seconds. The number of concurrent users was gradually increased, and that produced a stable performance rate until the point when the number of users was 135, a point where the average duration was 4.317 seconds. As the number of users was increased, the performance rate carried on degrading until a point where the average duration was 16.217 seconds for 409 concurrent users. This proposition is also supported by the statistical results in table 6.2.

- ***P_3:*** As more users (virtual and real) accessed the Web site in increasing numbers, the number of errors increased. The first batch of test experiments was free of errors. As the number of users was increased in the subsequent batches the number of errors increased. After all the test experiments were completed the total number of errors (from all the batches) was 302.

- ***P_4:*** This was merely an observation made by the test experiment facilitator. The number of real users tended to abandon the site as the performance started reaching an average duration of 13 seconds. The real users were not told, and were also not aware of the fact that the generally acceptable maximum limit for a performance rate is 10 seconds. The number of real users who abandoned the Web sites increased as the number of users (mainly virtual) increased. This was a result of the increased number of virtual users, which resulted in a decrease in the performance rate. Some of the users even asked the experiment facilitator if they should carry on trying or just quit. This observation made it clear that if it was a different setting where they were not sat down for test experiment purposes; they would have probably just abandoned the Web site without any hesitation. The interesting thing in this case was that this was observed with a small number of real users. It would be more interesting to realise what would be the case with a larger number of users.

- ***P_5***: In chapter two the term workload testing was defined as the process by which it is determined whether the system's resource allocation mechanism functions correctly, and how the system behaves under particular loads. As already discussed above, the Web site's performance rate decreased as the number of users was increased. At high levels of users the Web site could not handle the workload, and thus the performance rate started deteriorating rapidly when the number of users was increased to 135. This illustrated support for this proposition.

- ***P_6***: As the number of user was increased, and the performance rate decreased, the time spent on each function increased as a result of poor response time, and the real users experienced some difficulty in instantly accessing the Web site, which was due to line congestions. Thus this proposition was also supported.

- ***P_7***: It has been already stated that the generally acceptable levels of performance rate is a minimum of 3 seconds, and 8-10 seconds as the maximum for an average Web site. In figures 6.2 and 6.3 the observation is that as the performance was decreasing, there was a change in the acceptable levels of performance, and this change is directly linked to the decrease of the performance rate. The performance rate moved more and more towards the unacceptable levels when the number of

independent variables (number of users) was increased. Again, this proposition was supported.

After a discussion of all the above propositions, it was then possible to check and state whether or not the overall proposition is supported. Below is a discussion of the overall proposition.

Excluding P_1, which was not supported by the test results, all the other above propositions were supported by the test results. Thus it can be stated that the overall proposition is also supported by the test results. Overall, both figures 6.2 and 6.3 illustrate that the overall proposition is supported by the results. The two figures (6.2 and 6.3) illustrated that as the number of users increased, the performance of the Web site decreased. The performance stabilised at about 4 seconds until the number of users reached 120. It started moving up, decreasing further when the number of users was 135 and 142. From hence forth, the performance level sharply decreased until it reached 7.0167 seconds, where it stabilised again until 280 users, a point where the performance level decreased to 7.633 seconds. After that point, the performance levels saw a steady decrease until the highest average duration point of 16.217 seconds, where the number of concurrent users was 409. This supported the overall proposition.

The increase in the number of users resulted in a similar decrease in both response times for lower customer populations, and a proportional drop in customer throughput for a large population. The customer response time had a stronger sensitivity around 340 users, which is then reduced for a fully saturated system. As already stated above, this behaviour may be due to the limited server threads controlling the queue length at the database.

On the other hand, the 50% increase in the CPU demand of the shopping cart produced a very small effect on end-to-end performance results for the users. This was not particularly surprising, since there is a shopping cart for each active user, and the processor resources they run on were not saturated in the base case (less than 20% busy). The change produced a slight increase in the performance levels.

Images which were already loaded to the client's cache did not produce extra hits and therefore had no additional stressing effect on the system.

When the last batch of tests were run, the total number of concurrent users only increased to 409, with 26 errors (of the last batch) recognised by Web Performance Trainer.

These results were not surprising. The Web site owners stated that the Web site was never tested so extensively for performance in the operational environment, and thus they based the estimated number of concurrent users on the tests which were run on the company's networks.
The results shown in table 6.1 were computed by Web Performance Trainer, and exported to a spreadsheet. These results illustrate the stage of the study, when the test results were collected. There were multiple customers with a three-second think time in between interactions with the Web site. As the customers were interacting with the Web site, its throughput started saturating with around 340 users. The performance rate was considered satisfactory up to the ten-second-control-limit, which was reached about the same point as the saturation. Increasing the number of users decreased the performance up to about sixteen (16) seconds, a high level of performance not very satisfactory to the users of the Web application. At this level bottlenecks were experienced. Analysis of the results showed that the customers were queuing at the server task, which had only five (5) threads and that within the system the inventory manager was hundred percent (100%) saturated, mostly due to waiting for the database which was eighty percent (80%) busy.

In chapter one it was stated that one of the sub-goals of this study was *to investigate the possibility of predicting levels at which interventions could be necessary when performance has degraded beyond acceptable levels.* In order to illustrate the point when an intervention could be necessary, figure 6.3 was constructed in the form of a graph which illustrates a rolling total. This was done so that the point that is suggested as the point of intervention is representative of a range of values and not just a representation of just one test run. According to figure 6.3, the point at which an intervention could be necessary is when the number of users is about 340, a point where the maximum control limit of 10 seconds is reached. According to the graph,

the performance seems to remain stable and constant on ten seconds even when the number of users was increased. After the maximum control limit, the system could be said to be experiencing a bottleneck, and thus the need for intervention. If an analysis of the system could have been done on the server side, the testing tool could even pinpoint the specific resource which was causing the bottleneck. This was not extensively done in this study because it fell outside the scope of this study.

The subject Web site could perform better given some adjustments. The Web site could have enjoyed a better performance and user retention if most of the simple validations were done on the client side instead of the server side. Users spent lots of time on each function only because even the simplest of input validations happened on the server side. This caused an unnecessary build up of queues on the number of users 'stuck' in the same function, and increased time slicing.

The above results and analysis reveal that performance has an effect on the success or failure of Web-based e-commerce systems. It also revealed that testing performance on operational environments could contribute significantly to the awareness of whether or not a Web-based application should be optimised.

6.9 Faults and Errors Generated

Web Performance Trainer reported a total of three hundred and two (302) errors, which were generated by the Web site. This is a figure representative of the total number of errors generated by the total number of test batches as illustrated in Appendix E. As a result of the figure being too high, an inspection was conducted, and Web Performance Trainer showed that a large chunk of the errors were simply objects in the form of images which could not be displayed by the system. To go an extra mile in the analysis of the errors, one could activate the seeded faults individually, and determine which faults were revealed by which test cases. Such a step would require one to run an equal number of each test suite. This was not done in this study as it fell outside the scope of this work.

At the end of the test experiments, the real users were given a questionnaire for the purposes of exploring the things that real users of the Web application could express after such an experiment (see appendix A for the sample questionnaire). The questionnaire had to be filled in by the users. The exploratory and inconclusive results of the questionnaire are discussed below.

6.10 Discussions of the Post-Test-Session Assessments Where the Exploratory Questionnaire was Used

As stated before, the questionnaire was used merely for exploratory purposes. It is exploratory in nature because of the restricted sampling of the users and respondents and because the evaluation of the Web site was at a particular point in time. As a result of the exploration of the expressions by the real users, not much work was done regarding the questionnaire. These initial and inconclusive findings drew attention to satisfaction and culture issues associated with Web usability in the global e-commerce marketplace. Generally, the users appeared to be dissatisfied with the poor performance experienced when the sites were at the peak of their performance, as they displayed at lot of frustrations when the performance became poor. This could be due to the fact that degradation was experienced at unexplained intervals without the user knowing what was going on. This was the case on the Web site because it would not inform the users that it was busy processing.

Site abandonment and need for better performance appeared as being the most prevalent choice of most of the real users. The prevalence of occurrence confirmed the discoveries of the writers mentioned in chapter three and chapter five, that when the performance of a Web site degrades, the (frustrated) users are more likely to abandon the Web site. What was interesting was that the users did not really know what was an acceptable response time when using the Internet, but yet they knew when it was time to leave a site because of degraded performance. It would thus seem that e-commerce Web site users are mainly likely to stay on a site as long as its performance is within acceptable ranges.

There still seem to be much lack of tolerance of a poorly performing Web site. Despite appearing as being ignorant of the amount of time one is expected to spend waiting for a site to load up, users seem to be quick in judging a site that had unacceptable performance. This could either be attributed to the expectation of an e-commerce Web-based site to perform just like an HTML only Web site; or general comparisons with other known Web sites that perform better.

Although a fair number of the respondents indicated that they were unhappy with the Web site's performance when purchasing the products online, they generally indicated that they occasionally bought products and services online.

A significant number of the respondents indicated one or more visits to the same site they initially abandoned because of unacceptable performance. In future studies attention could be given to trying to discover the reasons for returning to sites initially abandoned, and attempt to compute the likelihood of a customer returning to any other site they abandoned. Some of the respondent did not know what the connection was between the performance and the number of users connecting to the same site.

It was also realised that Web site abandonment was not strictly a result of workload and high levels of stress on the system. There seems to be a relationship between the system performance and functional complexity. That is, as an application's functional complexity increases because of the nature of the input, its process capability decreases. The real users revealed that the variation in system's performance is not static. A system performance does not decrease strictly because of workload. Rather, the performance degradation was also related to the functional complexity of the task.

6.11 Suggested Actions to take in Improving Website Performance

Testing for remedial configurations for better performance and capacity was considered beyond the scope of this study. This requires conducting many iterations which was beyond the scope of this dissertation. To increase the performance of Web sites and the Web server an organisation should rely on load balancing so that each Web site or different Web servers perform optimally. Web server load balancing of IP-based applications can add a significant performance boost to Web applications (Gartner, 2001). However, before rushing to implement load balancing, managers should think first about the administrative costs involved when comparing this solution with other alternative technologies for increasing Web performance. Gartner (2001) suggested that load balancing is but one method for improving Web server performance. Other strategies include expanding Internet bandwidth, using faster network equipment, and designing more efficient Web applications, as well as optimising and upgrading Web server software and hardware.

However, the most popular alternative to load balancing is Web caching. Web caches are special servers that hold the data most often requested by users. Web servers are set to automatically refresh the caches any time data on the server is changed, so cache content is always current. With content being served from the cache rather than the Web server itself, the stress and load on the server can be decreased significantly, and improve the performance on the client/user side.

According to Gartner (2001), solutions are available to produce high availability, but organisations should be wary of the hype surrounding the issue. IT managers should be aware that good operational procedures can have an enormous effect on the difference between theoretical availability and the actual availability times touted by solution providers. Downtime costs a lot of companies large sums of lost revenue; and thus the need for awareness on how systems performance in a practical sense, and not just theoretically.

Bhattacharyya, Diot, Jetcheva and Taft (2001) wrote in their *Traffic Dynamics* report that enforcing load balancing could help in handling traffic demands on the Internet. They also wrote that in order to realise effective load balancing on the Internet, it is necessary to split traffic over multiple alternative paths. The load balancing should be applicable over long time scales, such as a few hours, or even potentially throughout the day-time.

Another thing to do is design and develop the application to do more of the simple input validation on the client side. As already noted, the Web site performed poorly because almost all input validations were done on the server side. If the Web site is designed to do simple input validations on the client side, it would relieve and free up the server of unnecessary client-requests processing, thereby reducing server side time-slicing. This would also reduce the time spent by users waiting for responses from the server.

6.12 Lessons Learned During the Test Experiments

Some lessons learned in the process of conducting the test experiments included:

- every step of the software testing process is important;
- user involvement in Web based software testing is important;
- the issue of thoroughness of testing requires management to make a judgment call, which tends to be more effective if the testing manager or facilitator has a thorough understanding of user testing objectives and the testing results being achieved;
- the design of an application plays an important role in how it would perform in its operational environment, and thus the need to design for optimum performance.

Web site success is critical to the establishment of viable Internet based business. Business-to-consumer (B2C) Web sites are particularly challenging in the performance-testing phase. This research used experiments to test performance of e-

commerce Web-based applications. Palmer (2002) wrote, in his analysis of software complexity metrics, that there is a challenge in the search for general measures with regards to Web site characteristics, performance, and usability. However, through the use of automated performance testing tools and software testing principles, this research has helped in identifying and validating measures of testing performance of e-commerce Web-based applications.

The tests revealed that properly planned tests, and well thought of, and evaluated automated testing tools can help in answering the question of whether or not a company is ready to trade online without losing revenue due to poor Web site application performance.

The results support better performance with a set of stated requirements that reflect on work in e-commerce, which suggests a focus on Web-based application performance. Low levels of performance also reflected in the results, with responsiveness being related to Web site success.

6.13 Chapter Conclusion

In this chapter, the work by Azizi (2002) and Weyuker and Vokolos (2000) theoretical foundations, about the effect of increased workload and stress levels on e-commerce Web-based applications, were applied in practice. This study provided further evidence for the usefulness of performance testing in real life, field conditions, where a Web site was tested in its operational environment.

The analysis and the discussions of the results of the test experiments conducted for this study proved to be addressing the main goal and sub-goals of this research. Except for just one (P_1), the propositions made in this study were also supported by the generated test results. The next chapter concludes this study by looking, among other things, at the directions for future research, lessons learned from this research, threats to validity, etc.

Chapter 7: Conclusion

7.1 General

This chapter presents a conclusion and a summary on the report of the results presented in the previous chapter. Limitations of this research as well as recommendations are discussed. Directions for future research are also discussed. This research focused on testing performance for e-commerce Web-based applications through the application of testing plans, testing tools and testing methodologies. Glass (2001) wrote that regardless of whether Web development is a new discipline, it certainly has not benefited from more mature testing and quality practices. It may be the fastest turnarounds that Web development requires, but there is a lack of testing experience of those involved in developing Web applications. In this study, this area appeared to have offered some interesting challenges which could be explored in the future. The scarcity of Web application testing methodologies led to a comparison of the currently available software testing plans and methodologies, as well as automated testing tools. This was done as part of the fulfilment of sub-goal five which was stated in chapter one.

The performance test experiments of the Web site showed that the Web-based e-commerce applications are still in the transition phase. Through proper testing, by experienced testers using sophisticated tools, the site stand to break out of the stigma mould which became a norm of most Web sites; namely, that a Web site will from time to time be down, and display poor performance. This can be attained by having e-commerce Web-based applications developers becoming active participants in the performance testing research activities. Since the time the tests were conducted, and the results made available to the Web site owners, the focus has been shifted from neglect to total awareness of performance testing and measurement effort to uplift the disadvantaged sections of the Web site, thus resulting in a whole new attitude toward the application testing on the Web sites. One can therefore conclude that the tested Web site is moving away from the traditional treatment of Web applications performance neglect, and is well on the way to offering online shoppers a better shopping experience and satisfaction. Much more work remains to be done in the

investigations of relations between online shoppers satisfaction and Web site performance. The work done in this study on that regard was merely exploratory and thus the number of users could not really serve as a spring board from which conclusive remarks could be made.

There is also a need to address the improvement of customer service, a hot issue and integral part of today's Customer Relationship Management (CRM) initiatives. CRM did not form part of the scope of this study, and was therefore not explored. Nevertheless, addressing the issues associated with the performance of e-commerce Web sites would help in cutting down on the number of customers who abandon Web sites due to poor performance.

7.2 Lessons Learned from this Study

7.2.1 Reproducible Results

No matter how performance testing is conducted, someone will scream "Foul Play!" (Merrill, 2002). The tests conducted in this study could be easily duplicated by anyone else who wanted to try. Depending on the environment and system configurations, which could differ from place to place, there is no doubt that the numbers might not be exactly the same as in this research.

Despite the fact that the Web technology is relatively new, this study has observed that even though the Web technologies have presented the software engineering community with new challenges, some aspects of software testing have not changed. This refers to the fact that software testing is a process that needs proper planning and also needs to be run in an organised and systematic manner. A large amount of time was spent on defining the parameters of the experiment in chapter five, where the plan and the methodology for testing the Web site had to be finalised, and thus avoiding simple mistakes when the test experiments commenced.

7.3 Theoretical and Practical Contributions of this Study to the e-Commerce and Web Development Disciplines

The major practical contribution of this research is that it conducted performance test experiments on a Web site which was already in its operational environment, and thus the results were more realistic and reflective of a real world setting. The lack of a wide pool of e-commerce Web application testing guidelines is a major weakness in the performance testing literature, and this research demonstrated a systematic approach to testing and improving such guidelines. This was achieved through an analysis of a number of different and varied testing plans and methodologies.

This study has also contributed to the literature of e-commerce Web-based application because it could also be used in assisting practitioners learn how to analyse the effectiveness of different performance testing strategies, testing tools, and methodologies. The analysis of the results provided clear directions for future research, as well as identifying problems in the investigations of software performance testing of e-commerce Web-based applications. More generally, the testing approach used in this study could be applied to validate and improve any quality guidelines meant for use on e-commerce Web sites.

7.4 Discussions on the Findings of this Study

The test tool was easy to understand, and it provided an accurate report on the analysis and statistics of the performance of the application of the Web site. It was possible to point out inconsistencies in the tests results. The results of the test experiments were an output of a well-run automated testing tool that met the main goal and sub-goals which were defined in chapter 1.

The findings of this study relate to some of the many aspects of the Internet business which could contribute to a better understanding of e-commerce, and thus the crafting of more successful online performance testing strategies.

It could also be concluded that e-commerce Web-based applications, which have been thoroughly tested for performance, offer the potential for companies to foster customers' loyalty in their Web sites, which can translate to higher profit growth. Hence, it would be wise for companies to invest in building up their online testing plans and methodologies to reap long-term growth from their online business.

7.4.1 Threats to Validity and Limitations

This study, like any other, had some limitations. Some primary limitations that could have influenced the results were identified. Some of these limitations were a result of using controlled experimentation. However, the advantage of control is more certain knowledge of causality (Elbaum, Karre and Rothermel, 2003).

Firstly, this study was conducted only on e-commerce Web-based applications. Web-based applications on other systems could respond differently from those on e-commerce environments. Thus, the results could be claimed to generalise only to Web applications using similar technologies. Additional Web applications should be studied to overcome these threats.

Secondly, there was a need to test the application for performance, with some of the focus directed onto the workload of the Web site, so as to evaluate the number of concurrent users the site could handle within the generally acceptable quality control limits. Although real users were preferred, obtaining a large number of real users of up to a thousand was not feasible. As a consequence, virtual users were opted for as they could be used to simulate common users tasks and activities. The risk of using virtual users who cannot give feedback is still a threat to validity. For example, only the real users could fill in the exploratory questionnaire at the end of the experiments. Nevertheless, however, the benefits of using an automated tool, using virtual users, were discussed in chapter 5.

Thirdly, the testing technique employed user interactions, which implied that one required the collection of data from users interacting with the e-commerce Web site being tested. The user navigation and buying patterns were not the focus of this study, but there was a wish to reproduce the activities users might perform in this type of a site. The instructions to the participants (real users) included a task and incentive to

make the experience more realistic for a set of customers, but were still merely an approximation of reality and constituted a threat to validity, because of representation on the part of the virtual users. The impact of this threat was diminished by providing a common strategy for all participants to select their input values.

Fourthly, the types of clients used, as well as the servers that hosted the applications, may change over time – the performance of the server could change dramatically if enhanced. Ideally, the study would be repeated over a period of time intervals to study how specific results change. As a consequence of these limitations, the results should not be viewed as conclusive in terms of the relative Web site performance. However, conclusions are drawn about the performance of the systems which were subjected to the test experiments.

Finally, this was an exploratory study. Thus, therefore, the statistical data analysis methods used in this study were not that strong on the statistical side. Also, if it meant using strong statistical data analysis methods, the propositions would have been called hypotheses, and that would have required the use of very strong statistical data analysis methods. Future studies could turn the propositions into hypotheses and also apply strong and better statistical methods which would not be as less restrictive as the propositions used in this study.

Despite these threats and limitations, this study should be used by software testers, software engineers, Web developers, and quality assurance managers to evaluate Web based applications by carrying out similar studies from a cross-section of their own.

7.5 Directions for Future Research

This study is not seen as the end but part of a beginning. The preliminary results of this study provided a foundation upon which future research could be conducted on Web-based application testing and e-commerce Web based applications. Future research could address the application of the results from this study to additional sites and with other tools. Testing plans might be expanded to include additional plans and methods for testing and measuring performance. Future work might also address differences in Web site task types. The findings from this study can be validated by future research using a larger sample over a longer period. Future research may also be able to identify more dimensions of the Web-based application performance testing strategies proposed by Nguyen (2000), which may be adopted by companies. Undoubtedly, there will be issues keyed to specific sites, and testers would test performance directly with more users, and over a longer period. The test experiments reported in this study represent an alternate, more generalised approach to testing the performance of e-commerce Web-based applications. The research contributes a set of results with specific analysis and discussion that could help the continuing process of improving Web-based application testing. Future research extending the current or exploring new dimensions of the e-commerce Web-based applications could provide further implications for both practitioners and researchers in their search for the most effective performance testing strategy congruent with their business. Some future studies could even target and categorise Web sites based on their offering and their target audience. The following is a list of some questions that may be pursued in the future:

- how often do requests to root servers fail due to timing out?
- are there correlations between particular domains and number of servers contacted?
- are there any common sources of misconfiguration with respect to domain delegation, and how often do they occur?
- what is the typical distribution of response time at a site?
- how many servers are typically contacted in order to load a single Web page of an e-commerce Web site?

- what fraction of name lookups during the user's browsing session are for non-cached names?
- what methods and guidelines should be used to ensure that a representative sample of Internet performance has been obtained?
- how much does the performance vary across Internet Service Providers in the same location?
- how effective are the testing techniques used during the design, development and implementation phases/stages of application development?

7.6 Concluding Remarks

In retrospect, the elements that led to the poor performance of the Web site were determined. Firstly, on the successful part, the Web site generated content, in a manner attractive and desirable to the user, was able to properly display requested data on screen when the Web site was at a point of experiencing difficulties. The real users were aware that they needed to wait for the Web pages to download, with the anticipation of requested data to be finally presented on screen even when the Web site started experiencing high loads. Secondly, on the failure part, the Web site failed to inform the users when it was experiencing high workloads and stressful conditions. This led to the Web site taking very long to load, and some users abandoning it.

Clearly, more detailed and focused work needs to be conducted to ensure that the impact of performance and e-commerce awareness can be maximised. One of the limitations of the study was that it did not investigate in detail the questionnaires of the respondents to determine the full extent to which poor performance of e-commerce Web-based applications affected the satisfaction levels of the users. This would be an area of future research which would help place the statistics within the context of actual e-commerce activity and how better performance could work towards developing e-commerce on a wider scale.

This study has also presented a different approach for testing e-commerce Web-based applications. This approach differed from existing approaches in that live data was captured to generate test cases, leading to a reduction in the amount required for test case generation by the tester, and made use of these test cases to run performance tests on a live Web site. Further more, the results of a controlled experiment indicated that this approach's effectiveness was comparable and likely complementary to more formal white box testing approaches.

Since the study of e-commerce has many dimensions, it is hoped that this study will open or contribute to research initiatives in the future. The results of these experiments suggest several different directions for future work. Firstly, the combination of traditional testing techniques and modern techniques would seem to possess a potential that the software engineering community has not been able to fully exploit. Secondly, the analysis of the test results suggested that using a larger number of real users would provide more reliable results because the interaction by real users with the services of a Web site are more realistic compared to those by the virtual users.

This study also addressed the use of testing plans and methodologies for testing performance on the Web. The results are encouraging. The test experiments showed a considerable resemblance to the models of software testing adopted in chapter five. The results suggested that:

- the presently developing Web technologies need more research into how it can be better tested; and
- automated testing, like the one used in this study, would be feasible for testing Web-based e-commerce applications. When using an automated testing tool one can simulate thousands of users, compared to the almost impossible task of sitting down thousands of real users.
- there is a need to evaluate the effectiveness of the testing techniques used to test Web-based applications.

The main goal of this study was to investigate the ways and techniques used to test e-commerce Web-based applications, with particular attention on performance testing, and conducting experiments using those techniques, analyse and discuss performance

testing results, draw up conclusions and document the whole testing processes and procedures followed during the experiments. This was done throughout the previous chapters.

In addition to the main goal, the study also had sub-goals which also needed to be addressed. The first sub-goal was to investigate the foundations and current research issues on software testing. This was addressed in chapter two. The second sub-goal was to investigate the main issues in performance testing and measurement. Chapter two and chapter four addressed this sub-goal. The third, and fourth sub-goals were to investigate the role of testing on Web-based application development, and to research the issues of performance testing on Web-based application used on e-commerce Web sites. Chapter three was dedicated to addressing these sub-goals. The fifth sub-goal was to list, compare and choose an automated Web testing tool from the list. Chapter five addressed this goal, where three different automated Web testing tools were listed, compared and the one with the highest score was employed in the experimental tests of this study. Chapter six addressed the sixth and the seventh sub-goals, where different testing plans were analysed, adopted and applied in the test experiments, propositions were defined and checked whether or not they were supported by the test results, the test results were properly documented, analysed and discussed.

The activities that were carried out throughout this study focused on investigating and addressing the goals of this study. This study has presented an investigation into the performance testing on e-commerce Web-based applications, which clearly addressed the main goal and the sub-goals as originally stated in chapter one.

There is no expectation that this research will be the authoritative, last work on the testing of e-commerce Web-based applications. Any given application will stress different parts of a system. Some application will run better on some servers than others. This research provides data that helps other researchers and practitioners weigh the performance trade-offs of various Web sites.

References

ApTest. Software Testing Glossary. Available from: www.aptest.com. [Accessed 12 July 2002].

Arthur, L. J. 1988. Software Evolution: *The software Maintenance Challenge.* New York: John Wiley & Sons.

Asymetrix, Inc. *Software Testing.* Available from: www.asymetrix.com [Accessed 20 May 2002]

Authorware Resource Page: *Shockwave Examples Files.* Available from: www.prenhall.com/divisions/ESM/app/hooper/shocked/examples.htm . [Accessed 23 August 2000].

Azavar Technologies. *B2C eTailing (Business to Consumer e-Commerce).* Available from: http://techlibrary.insurancetech.com/data/. [Accessed 2001 January].

Azizi, A. H. Web Application Testing: *Defining Load Testing.* Available from: www.veritest.com/testersnetwork/index.htm. [Accessed 12 July 2002].

Bach, J. 2001. Satisfied Test Planning Guide: Building the Plan (Draft v0.3). Available from: www.satisfice.com/tools/build-the-plan.pdf. [Accessed 20 July 2002].

Bach, J. Explaining Testing to Them - *Helping non-testers understand and support your work.* Available from: www.stqemagazine.com/featured.asp?id=19. [Accessed 17 August 2002].

Bacheldor, B. Push for Performance. Available from: www.informationweek.com. [Accessed 15 September 2001].

Becker, S. A. 2002. An Exploratory Study on Web Usability and the Internationalization of US e-business. Journal of Electronic Commerce Research, Vol.3, No.4, 2002: p.265.

Bhattacharyya, S., Diot, C. Jetcheva, J. and Taft, N. 2001. Pop-level and Access-Link-Level Traffic Dynamics in a Tier-1 POP. Proceedings of the First ACM SIGCOMM Internet Measurement Workshop (IMW 2001). San Fansisco, Califonia, USA, 1-2 November 2001: pp.39-48.

Bowerman, B. L., O'Connell, R. T., and Hand, M. L. 2001. Business Statistics in Practice. 2nd Edition. New York. McGrawl-Hill/Irwin.

Brink, T., Gergle, D. and Wood, D. S. 2002. Designing Web Sites That Work: *Usability for the Web.* USA: Morgan Kaufmann Publishers.

Brown, A. W., Christie, A. M. and Dart, S. A Case Study in Software Maintenance. Available from: http://www.sei.cmu.ed/lgacy/scm/abstracts/abssw_maint_case_study_TR08_93.htm. [Accessed 29 May 2000].

Carr, M. Managing Software Maintenance with Metrics. Available from: http://ww.softdim.com/psqt99north/. [Accessed 13 July 2001].

Chang, C. H., and Hsu, C. C. 1999. Enabling Concept-Based Relevance Feedback for Information Retrieval on the WWW, IEEE Transactions on Knowledge and Data Engineering, August 1999, Vol.11, No.4: p.595.

Chapman, C. The Perils of e-Commerce Security. Available from: http://www.veritest.com/testersnetwork/index.htm. [Accessed 16 September 1998].

Cheng, A. M. K., Clements, P. and Woodside, M. 2000. Guest Editors' Introduction: *Workshop on Software and Performance*, IEEE Transactions on Software Engineering, December 2000, Vol.26, No.12: p.1121.

Chevalier, J. L. and Quaterman, J. S. Beyond Mere Latency. Available from: http://www.matrix.net. [Accessed 16 October 2002].

Chillarege, R. 1999. Software Testing Best Practices: IBM Research. Available from http://www.ibm.com. [Accessed 26 April 2003].

Chmura, A. and Sharon, D. 1996. Tools Fair: *Untangling the Web with Web and Client/Server Development Tools*. IEEE Software September 1996: p.62.

Cho, S., Byun, F. H. and Sung, M. 2003. Impact of the High-speed Internet on user Behaviour: *case study in Korea, Internet Research.* Electronic Networking Application and Policy, Vol.13, No.1, 2003: pp.49-59.

Cohen, D. M, Dalal, S. R., Fredman, M. L. and Patton, G. C. 1997. The AETG System: *An Approach to Testing Based on Combinatorial Design.* IEEE Transactions on Software Engineering, Vol. 23, No. 7, July 1997: p.437.

ComCity: *Terms Glossary*. Available from: http;//www.comcity.com. [Accessed 03 August 2000].

Communications of the ACM: *Learner-Centred Design.* Vol. 39, No 4, April 1996.

Communications of the ACM: *PUI/Programming by Example.* Available from: http://www.acm.org/cacm. [Accessed 03 August 2000].

Communications of the ACM: *PUI/Programming by Example.* Vol. 43, No 3, March 2000.

Constantine, L. L., and Lockowood, L. A. D. 1999. Software for Reuse: *A practical Guide to the Model & Methods of Design of Usage-Centred Design,* New York: Addison-Wesley.

Daily Build and Smoke Test, IEEE Software, Vol.13, No.4, July 1996, Available from: http://www.computer.org. [Accessed 17 August 2002].

Daily Build and Smoke Test. Best Practice. IEEE Software, Vol. 13, No. 4, July 1996.

Daily Mail & Guardian. Icon of our Age Turns 20. August 2001.

Dalgleish, J. 2000. Customer-Effective Web Site. California.

Deja. Graphics and Multimedia Software: *Macromedia Authorware Attain.* Available from: http://www.deja.com/products/at_a_glance/glance.xp. [Accessed 23 August 2000].

Delamaro, M. E., Maldonado, J. C., Mathur, A. P. 2001. Interface Mutation: *An Approach for Integration*, IEEE Transactions on Software Engineering. Vol. 27, No.3, March 2001: p.229.

Delamaro. 2001. Interface Mutation: *An Approach for Integration Testing.* IEEE Transaction on Software Engineering. March 2001, Vol. 27, No. 3: p228-229.

Dennis, A. and Wixon, B. H. 2000. Systems Analysis and Design – *An Applied Approach.* USA: John Wiley & Sons Inc.

Donnelly, D. 1997. WWW Design: Web *Pages from the around the World.* Massachusetts: Rockport Publishing, Inc.

Dromey, R. G. 1996. *Concerning the Chimera.* IEEE Software, January 1996: pp.33-34.

Dustin, E., Rashka, J. S. and Paul, J. 1999. Automated Software Testing: *Introduction, Management, and Performance.* USA: Addison-Wesley.

Eisermann, M. 2002. Technical Research Paper. Performance tests with the Microsoft Internet Security and Acceleration (ISA) Server, Bad Aibling, Germany.

Elbaum, S., Karre, S., and Rothermel, G. 2003. Improving Web Application Testing with User Session Data. 25th International Conference on Software Engineering (ICSE 2003), Portland, Oregon, 3-10 May 2003: pp.49-57.

Elliot, S., and Fowell, S. 2000. Expectactions versus Reality: *A Snapshot of consumer experiences with Internet retailing,* International Journal of Information Management, Vol. 20: 323 – 336.

eMakerter, Inc. Available from: http://www.emarket.com. [Accessed 12 July 2002].

Empirix. Top 25+ Reasons Web Applications Don't Scale. Available from: http://www.empirix.com. [Accessed April 2003].

Filos, E. and Ouzounis, V. K. 2002. Virtual Organisations: *Technologies, Trends, Standards and the Contribution of the European RTD Programmes,* International Journal of Computer Applications in Technology, Special Issue: "Applications in Industry of Product and Process Modelling Using Standards, 2002.

Forsyth, I. 1998. Teaching and Learning Materials and the Internet. London: Kogen ltd.

Frank Cohen. Web Services. Available from: www-106.ibm.com/developerworks/webservices/library/ws-testsoap/#h3. [Accessed 04 August 2002].

Fraternali, P. and Paolini, P. 2000. Model-Driven Development of Web Applications: *The Autoweb System,* ACM Transactions on Information Systems, October 2000: p.324.

Furht, B. 1999. Handbook of Internet & Multimedia Systems and Applications, USA: CRC Press LLC.

Gao, J., Chen, C., Toyoshima, Y., Kung, D. and Hsia, P. 1996. Identifying Polymorphism Change and Impact in Object-Oriented Software Maintenance. Available from: http://www3.interscience.wile.com/abstracts. [Accessed 29 May 2000].

Gartner. Network Load Balancing in a Windows Data Centre: *Perspective.* Available from: http://www.gartner.com. [Accessed December 2001a].

Gartner. Network Load Balancing in a Windows Data Centre: *Perspective.* Available from: http://www.gartner.com. [Accessed 17 December 2001b].

Gartner. Understanding the New Trends in Project Management. Available from: http://www.gartner.com. [Accessed 21 March 2001c].

Glass, R. L. 1998. Maintenance: *Less Is Not More.* IEEE Software, July/August 1998: p.67.

Glass, R. L. Maintenance: *Less is not more.* IEEE Software, July/August 1998.

Gold Star Web. Web Maintenance. Available from: http://www.gs-web.com. [Accessed 31 July 2000].

Gordon, R. and Gordon, R. 1998. Information Systems: *A Management Approach*, 3rd Edition. New York: Wiley.

Graham, D. and Fewster, M. 1999. Software Test Automation: *Effective Use of Test Execution Tools*, New York: Addison-Wesley.

Graja, H. and McManis, J. 2001. Quantifying Customer Satisfaction with E-commerce Websites. 17th IEE UK Teletraffic Symposium, Dublin, Ireland, May 16-18, 2001.

Gross, K. Web Performance Monitoring Deemed "Crucial". Available from: http://www.newmedia.com. [Accessed 13 October 2001].

Grossman, D., McCabe, C. M., Staton, C., Bailey, B., Frieder, O. and Roberts, D.C. 1996. Performance Testing a Large Finance Application, IEEE Software, September 1996: pp.50-53.

Grove Consultants. Testing Tool Information. Available from: http://www.grove.co.uk/tool_information/choosing_tools.html. [Accessed 30 October 2001].

Guide to the Project Management Book of Knowledge (PMBOK guide). 2000. Project Management Institute. USA.

Hagen, G. System Testing - *Mortgage Banking*. August 1998, Vol. 58, issue 11. p.97. Available from: http://ww.epnet.com/cgi-bin/epwnorb/key=oHFwnSQ. [Accessed 13 May 2001].

Hall, T. L. 1996. Utilising multimedia software ToolBook 3.0. USA: Boyd & Fraser publishing company.

Hamptonu. Software Testing Techniques. Available from: http://www.csc.hamptonu.edu/~cezzar/soften1/softtest.htm [27 August 2002].

Harrison, W. Software Testing. Available from: http://www.cs.pdx.edu/~warren/cs409 [Accessed 03 August 2000].

Hazari, S. 1998. Evaluation and Selection of Web Course Management Tools, The Robert H Smith School of Business, University of Maryland, College Park. Available from: http://sunil.umd.edu/documents/webtools/coursetools.htm. [Accessed 16 August 2000].

Heller, S. and Drennan D. 1997. The Digital Designer – *The Graphic Artist's Guide to the New Media.* New York: Watson-Guptill Publications.

Hetzel, B. 1988. The Complete Guide to Software Testing. 2nd edition. New York: John Wiley & Sons, Inc.

Hetzel, B. 2000. The Complete Guide to Software Testing: *Server Software Testing on the Desktop and the Web.* New York: Prentice Hall.

Hetzel, W. C. 1973. Program Test Methods. New Jersey. Prentice Hall.

Holmes, M. 2002. Web Usability and Navigation: *A Beginners Guide.* USA: The McGra-Hill Companies.

Horgan, J. R. and Mathur, A. P. 1992. Assessing Testing Tools in Research and Education. IEEE Software May 1992: p.61.

Horowitz, E. 1998. Migrating Software to the World Wide Web. IEEE Software Ma/June 1998: p.19.

Houdek, F. and Kempter, H. 1997. Quality Patterns – *An Approach to Packaging Software Engineering experience.* Software Engineering Notes. Vol. 22, No. 22 May 1997: p.81.

Hower, R. Software QA and Testing Frequently-Asked-Questions Parts 2. Available from: http://www.softwareqatest.com/qatfaq2.html. [Accessed 28 August 2002].

Hower, R. Web Site Test Tools and Site Management Tool. Available from: http://www.softwareqatest.com/qatweb1.html. [Accessed 28 August 2002].

Ideva. Internet Development Associates. Available from: http://www.ideva.com/mits/mits_02.htm. [Accessed 27 August 2002].

IEEE MultiMedia. Available from http://computer.org [Accessed 15 August 2000].

Jain, R. 1991. The Art Computer System Performance Analysis. Wiley: New York.

Johanna Rothman, J. Release Criteria: Is This Software Done? How to know if your software is ready to release. Available from: http://www.stqemagazine.com/featured.asp?id=21. [Accessed 27 August 2002].

Johnson, J. R. 1991. The Software Factory: *Managing Software Development & Maintenance, 2nd edition.* Wellesley: QED Information Sciences, Inc.

Kavi, K and Nahouani, E. 1996. Software Tools Assessment. IEEE Software, September 1996: pp.23-25.

Kemmerer, R. A. 2003. Cybersecurity. Proceedings of 25th International Conference on Software Engineering, Portland, Oregon 3-10 May 2003: p.705.

Keynote Systems. Available from:http://www.keynote.com/measures/top10.html. [12 July 2002].

Kitchenham, B. and Linkman, S. 1998. Vaildation, Verification, and Testing: *Diversity Rules.* IEEE Software, July/August 1998: p.47.

Klaasedv: Available from: http://www.klaasedv.com/article/category3.htm. [Accessed 28 August 2002].

Knowledgestorm. Four Elements of Online Success. Available from: http://www.knowledgestorm.com. [Accessed August 2002].

Kolawa, A. 2000. Automatic C/C++ Unit Testing. IEEE Software March/April 2000: p.88.

Koloski, D. Implementing an Effective Web Application Testing Process. Available from: http://www.empirix.com. [Accessed April 2003].

Larson, G. B. 1995. The User Acceptance Testing Process: A Case Study. Journal of Systems Management, September/October 1995, Vol. 46, No.5: pp.1-5

Laudon, K. C. and Laudon, J. P. 1998. Management Information Systems, 5th Edition. New Jersey: Prentice Hall International.

Lauesen, S. and Younessi, H. 1998. *Is Software Quality Visible in the Code?* IEEE Software Jule/August 1998: p.70.

Laursen, D. Data Marts and The Web. Available from: http://www.testersnetwork.com. [Accessed 20 July 1998].

Linberg, K. R. Software Developer Perceptions about Software Project Failure: *A Case Study.* Available from: http://www.lc.capellauniversity.edu/~klinberg/casestd_linberg.pdf. [Accessed 13 October 2001].

Littlewood, B., Popov, P. T., Strini, L. and Shryane, N. 2000. Modeling the Effects, of Combining Diverse Software Fault Detection Techniques. IEEE Transaction on Software Engineering, December 2000. Vol. 26, No. 12: p.1157.

Lootsma, F. A. 1999. Multi-criteria Decision Analysis via Ratio and Difference Judgement. Boston: Kluwer Academic Publishers.

Luqi, Chang, C. K. and Zhu, H. 1998. Specifications in Software Prototyping. Journal of Systems and Software, No.42, 1998: pp.125-140.

Marick, B. Classic Testing Mistakes. Available from: http://www.testing.com/writings/classic/mistakes.htm. [Accessed 28 August 2002].

Maybury, M. T. and Wahlster W. 1998. Readings in Intelligent User Interfaces. San Francisco: Morgan Kaufmann Publishers, Inc.

McDermid, J. 1993. Software Engineer's reference Book. USA: CRC Press, Inc.

Mehrotra, R. 2000. Advancing the Internet. IEEE MultiMedia, Vol. 7, No. 1, January – March 2000: p.9.

Melville, S. & Goddard, W. 1996. Research Methodology. *An Introduction.* Kenwyn: Juta & Co Ltd.

Merrill, C. L. Servlet Container Performance Report: *Comparing The Performance of J2EE Servers*. Available from: http://www.webperformance.com. [Accessed 11 December 2002].

Miller, E. 2000. Web site Loading and Capacity Analysis. Software Research, Inc, 2000.

Miller, E. Web site Testing. Software Research, Inc. Available from: http://www.soft.com/evalid/technology/white.papers/website.testing.html. [Accessed 12 December 2000].

Mochal, T. Hammer out Your Tactical Testing Decisions With a Testing Plan. Available from: http://www.techrepublic.com. [Accessed 27 August 2001].

Mochal, T. Keep the Software Testing Life Cycle Spinning. Available from: http://www.techrepublic.com. [Accessed 27 August 2001].

Mochal, T. Think Ahead: Defining Your Testing Strategy. Available from: http://www.techrepublic.com. [Accessed 27 August 2001].

Mosley, D. J. 1999. Client Server Software Testing on the Desk Top and the Web. New Jersey: Prentice Hall.

Munson, J. C and Khoshgoftaar, T. M. 1992. Measuring Dynamic Program Complexity. IEEE Software November 1992: p.48.

Murata, T. 2003. Visualising the Structure of Web Communities Based on Data Acquire From a Search Engine. IEEE Transaction on Industrial Electronics, 50(5):860.

Murray, B. T and Hayse, J. P. 1996. Testing ICs: *Getting to the Core of the Problem.* Computer, November 1996: p.33.

Musa, J. 1996. Software-Reliability – *Engineering Testing.* Computer, November 1996: pp.61-66.

Myers, B. A., McDaniel, R. and Wolber, D. 2000. Intelligence in Demonstrational Interfaces. Communications of the ACM, March 2000, Vol. 43, No. 3: p.84.

Nguyen, H. Q. 2000. Testing Applications on the Web: *Test Planning for Internet-Based Systems.* New York. John Wiley & Sons, inc.

Nielsen. Available from: http://www.nielsen-netratings.com/. [Accessed 13 November 2002].

Nixon, B. A. 2000. Management of Performance Requirements for Information Systems, IEEE Transactions on Software Engineering, December 2000, Vol.26, No.12: p.1123.

Ntafos, S. 1998. On random and partition testing. Paper presented at ISSTA 98, Clearwater Beach, FL.

Ocampo, G. Testing Considerations for Web-Enabled Applications. Available from: http://www.veritest.com/testersnetwork/index.htm. [Accessed 02 September 1999].

Onoma, A.K, and Yamaura, T. 1995. Practical Steps Toward Quality Development. IEEE Software September 1995: p.68.

Osterweil, L. and Clarke, L. A. 1992. A proposed Testing and Analysis Research Initiative. IEEE Software, September 1992: p.90.

Ould, M. A. 1994. Strategies for Software Engineering: *The Management Risk & Quality.* New York: John Wiley & Sons, Inc.

Overview of Testing. Available from: http://www.thehathaway.com/testingV6.hml#TTS [Accessed August 2002].

Palmer, J.W. 2002. *Web Site Usability, Design, and Performance Metrics*, Information Systems Research. Vol. 13, No. 2, June 2002.

Parikh, G.1980. Techniques for System Maintenance. Nebraska: Erthnotech, Inc.

Patel, J., Schnecker, M., Desai, G., and Levitt, J. 2002. Tools for Growth in E-Commerce (part 1). InformationWeek. Available from: http://www.samarket.com. [Accessed 09 November 2002].

Perry, W. 1995. Effective Methods for Software Testing. New York: John Wiley & Sons, Inc.

Petroutsos, E. 2000. ASP 3 Instant Reference, New Delhi: BPB Publications.

Pettichord, B. Seven Steps to Test Automation Success. Available from: http://www.pettichord.com. [Accessed 03 March 2001].

Plan on Testing Success. Available from: http://www.softwaretestinginstitute.com/testplan.htm. [Accessed 12 July 2002].

Poston, R. M and Sexton, M. P. 1992. Evaluating and Selecting Testing Tools. IEEE Software May 1992: pp.33-41.

Potosnak, K. 1988. Human Factors: *Recipe for a Usability Test.* IEEE Software. November 1988: p83.

Powers, M. Why Test the Web? How Much Should You Test? Available from: http://www.veritest.com/testersnetwork/index.htm. [Accessed: 15 November 2000].

Ratajczyk, J. Testing Applications on the New Platforms. Available from: http://www.informit.com/articles/index.asp?st=41355 . [Accessed 23 August 2000].

Remenyi, D. and Grant, K. It was a shock when Boo went under: *The Legacy of the e-Buble – Lessons for Managers.* Journal of General Management. Vol 29, No 3. p24-36. Spring 2004

Sabin, P. Client/Server Performance Testing: *Defining the Key Criteria for Success.* Available from: www.veritest.com/testersnetwork/index.htm. [Accessed 12 July 2002].

Schmeiser. L. 1999. Text Editor Intro, On-Line Tutorial & Related Links. Available from: http://www.harrold.org/rfhextra/htmlhelp.html . [Accessed 23 August 2000].

Schneider, G. P. and Perry, J. T. 2002. Electronic Commerce. Cambrige: Course Technology.

Shelly, G. B., Cashman, T. J., Waggoner, G. A. and Waggoner, W. C. 1998. Discovering computers 98 – *A Link to the future.* Cambridge: International Thompson Publishing company.

Shelly, G. B., Cashman, T. J., Waggoner, G. A. and Waggoner, W. C. 2000. Discovering computers 98 – *A Link to the future.* Cambridge: International Thompson Publishing company.

Shen, V. Expert Systems. IEEE Software. November 1988.

SIGCSE Bulletin – inroads, ITiCSE Proceedings. Vol. 31, No 3. September 1999.

Singh, S. 1999. A Preliminary Investigation of Web-based Courseware Tools. BTech dissertation, ML Sultan Technikon, Durban, South Africa.

Singh, S. and Erwin, G. J. 1999. A Survey of usage of features of World Wide Web courseware tools. Fourth KwaZulu Natal Research Conference, Durban, South Africa.

Software Quality – *The Testing Lifecycle.* Available from: http://www.brs.net. [Accessed 15 September 2001].

Software Testing Techniques. University of South Australia, School of Computer and Information Science. Available from: http://louisa.unisa.edu.au/sel/testing-notes/week7a96.htm . [Accessed 12 August 2002].

Software Testing Techniques. University of South Australia, School of Computer and Information Science. Available from: http://louisa.unisa.edu.au/sel/testing-notes/week8_96.htm. [Accessed 12 August 2002].

Sommerville, I. 1997. Software Engineering. UK. Addison-Wesley.

Splaine, S. and Jaskiel, S.P. 2000. The Web Testing Handbook. USA: STQE Publishing.

Squires, G. L. 1985. Practical Physics. 3rd Edition, Cambridge University Press.

Stark, G. E., Oman, P., Skillicorn, A. and Ameele, A. 1999. An examination of the Effects of Requirements Changes on Software Maintenance Releases. Available from: http://www3.interscience.wile.com/abstracts. [Accessed 29 May 2000].

Swanson, E. B. and Lientz, B. P. 1980. Software Maintenance Management. , Phillipines: Addison-Wesley.

Systest: Software Testing Solutions to Make Your Project Soar. Available from: http://www.systest.com. [13 July 2001].

Tan, W. and Gable, G. G. Attitudes of maintenance personnel towards maintenance work: a comparative analysis. Available from: www3.interscienc.wiley.com. [Accessed 13 December 2000].

Tansely, D. 2002. Create Dynamic Web Pages using PHP and MySQL. New York: Addison-Wesley.

TechRepublic: The Project Office. Available from: http://www.techrepublic.com. [Accessed 03 August 2000].

Ten Tips for Getting Useful Information from Users. IEEE Software, July 1998: p89.

Test Focus. Automation: *An Introduction,* Vol. 3, No. 7, July 2002: pp.1-8.

Testing FAQs. Available from: Testingfaqs.org/t-design.htm. [Accessed 15 October 2002].

Thehatthaway. Plan Testing Activities. Available from: http://www.thehathaway.com/plantest.hml. [Accessed 27 August 2002].

Thompson, S. H., Teo, and Tan, J. S. Senior Executives' Perceptions of Business-to-Consumer (B2C) Online Marketing Strategies: *The Case of Singapore.* Internet Research: Electronic Networking Application and Policy, Vol.12, No.3, 2002: pp.258-274.

Tice, G. Real-time systems. IEEE Software, September 1988.

Turban, E., Lee, J., King, D. and Chung H. 2002. Electronic Commerce: *A Managerial Perspective.* New Jersey. Prentice-Hall.

Wakid, S. A., Kuhn, R. D. and Wallace, D. R. 1999. Toward Credible IT Testing and Certification. IEEE Software July/August 1999: p.39.

Web Connects Web Maintenance. Available from: Http://www.iserveyou.com/pages/maintenance.htm [Accessed 05 August 2000].

Web Metro Internet Services: *Web site maintenance and Services*. Available from: Http://www.webmetro.com/web_site_maintenance.htm [Accessed 05 August 2000].

Web site Design and Maintenance. Available from: Http://www.soft-options.co.uk. [Accessed 03 August 2000].

WebPerformance. Available from: www.webperformance.com. [Accessed 15 October 2002].

Weyuker E. J. and Vokolos, F. I. 2000. Experience with Performance Testing of Software Systems: *Issues, an Approach, and Case Study.* IEEE Transactions on Software Engineering, Vol. 26, No. 12, December 2000.

Weyuker, E. J. 1998. Testing Component-Based Software: *A Cautionary Tale.* IEEE Software, September/October 1998: pp.55-56.

Weyuker, E. J., Ostrand, T. J., Brophy, J. and Prasad, R. 2000. Clearing a Career Path for Software Testers. IEEE Software March/April 2000: pp.77-78.

Whelan, V. B. B2B Commerce: *The Next Frontier.* Available from: http://www.business2.com. [Accessed 13 November 2002].

Whittaker, J. A. 2000. What Is Software Testing? And Why Is It So Hard?, IEEE Software, January/February 2000: pp.70-77.

Whitten, J. L. and Bentley, L. D. 1998. Systems Analysis and Design Methods, 4th edition. USA: Irwin McGraw-Hill.

Wikipedia. Software Engineering. Available from: http://www.wikipedia.com/wiki/software_engineering. [Accessed 12 August 2002].

Wonnacott, T. H. and Wonnacott, R. J. 1990. Introduction to Statistics for Business and Economics, fourth edition. USA: John Wiley & Sons.

Yamaura, T. 1998. How to Design Practical Test Cases. IEEE Software November/December 1998: p.33.

Zona Research. Need for Speed. Available from: http://www.zonaresearch.com. [Accessed 13 May 2001].

Appendix A: List of Terminology Used in this Study

- **Acceptance Testing**

Meets user operational needs (klaasedv.com, 2002). According to ApTest (2002) acceptance testing is conducted to enable a user/customer to determine whether to accept a product. It is normally performed to validate that the software meets a set of agreed acceptance criteria.

- **Benchmark**

In computers a 'benchmark' is a test, or set of tests, designed to compare the performance of one computer system against the performance of others.

- **Black-box Testing**

Testing base on an analysis of the specification of a piece of software without its internal working. The goal is to test how well the components conforms to the published requirements (ApTest 2002).

- **Browser**

The generic term for software programs that allows a networked computer to connect to other Internet sites.

- **Compatibility Testing**

Compatibility testing is performed in order to verify that the product functions without difficulties or discrepancies due to incompatibility with a platform configuration. Tests are run on several different computer configurations that are considered the "Industry Standard".

- **Configuration Testing**

Configuration testing checks how the product works on different hardware and when combined with different third party software (Marick, 2002).

- **Conformance Testing**

The process of testing that an implementation conforms to the specifications.

- **Download**

The electronic transfer of information from a remote computer to a local one.

- **Electronic Commerce/e-commerce**

The transfer of business electronically rather than via paper. Also referred to as EDI. (Electronic Data Interchange).

- **Exhaustive Testing**

Testing which covers all combinations of input values and preconditions for software under test.z

- **Forms**

HTML-based specifications of widgets, which includes text fields, password fields, multilane text entry, checkboxes, toggles, option menus, multiple selection lists, images, hidden fields, and submit buttons.

- **Functionality Testing**

Functionality testing is performed to ensure that the product functions the way it was designed to according to the design specifications and documentation. Both positive and negative test cases are run to verify that the program responds in the correct manner. According to Systest (2001) functional testing is testing to ensure that all functions specified in the requirements and design are present and work as intended. If a system is not explicitly defined in documentation, functional testing will rely on the results of interviews and analysis of the system with key client personnel. Functional testing is testing the features and operational behaviour of a product to ensure they correctly conform to specifications (ApTest, 2003).

- **Gateway**

A device that acts as a connector between two separate networks. It has interfaces to more than one network and can translate the packets of one network to another, possibly dissimilar, network.

- **Host Computer**

In the context of networks, a computer that allows a user or user's client to connect to it, in contrast to a network server, which provides services to a user through a intermediary host computer.

- **Hyperlink**

A word or graphic in a file displayed on-screen with some form of highlighting (colour or underlining or both), which represents hidden text containing the URL of another document, which is displayed when one clicks on the highlighted word or graphic.

- **Install/Uninstall Testing**

This type of testing is performed to ensure that all install features and options function properly. This also ensure that the uninstall works properly by verifying that all of the installed directories and files are removed or correctly modified.

- **Installation Testing**

The purpose of installation testing is to validate that a system can be successfully installed on the target platform, that once installed, the system is usable, and that the guidelines used to direct installation are accurate (systest.com, 2001).

- **Integration Testing**

The goal of integration testing is to put the units in their intended environment and exercise their interactions as completely as possible (Delamaro et al, 2001). According to systest.com (2001) integration testing is the process of systematically piecing together units, objects, or modules of code in an attempt to approximate, as closely as possible, the working application and in the process to uncover missing links, calls, gaps in the system, and the obvious bugs.

- **Load Generator**

Something that provides part of a workload to a System Under Test, for a benchmark. Commonly this term applies to a "client" system that is used to drive system under test over a LAN. This term can also be used to describe a process (either on a 'client' or the SUT) which is generating a load for the benchmark.

- **Load level**

For any benchmark which submits various amounts of work to a System Under Test, a load level is one such amount of work. This is usually in terms of expected throughput.

- **Modem**

An acronym for MOdulator/DEModulator. It is a device that converts the digital signals in one's computer to analog signals, and vice-versa, to enable computer communication through analog telephone lines.

- **Multitasking**

Allows users to work with several programs at once without exiting any one program. The different programs are shown in different windows on the computer screen, and a user can work in different windows as needed.

- **Mutation Testing**

Mutation testing is an error-based testing adequacy criterion proposed by DeMillo et al, initially with the name "Mutant Analysis."

- **Navigate**

The process of moving about purposefully with a virtual environment. This term suggests the wide-open spaces involved in virtual environments and the need t find one's way with the help of navigational devices (documents, links, web maps, files, menus) that are particular to electronic environments.

- **Network Testing**

Because many of today's applications have multi-user capabilities, tests need to be run to ensure that a product is indeed networkable. Network testing makes sure that a product can be run on several network OS environments.

- **Operational Testing**

Testing in a "true end-user environment" following all business rules and processes. The goal of operational testing is to demonstrate that a system can be used to perform its jobs following the exact set of processes and steps that would be used by the target customer or end-user (systest.com, 2001).

- **Performance Testing**

After having published the Web site to the World Wide Web the it must be tested to discover if it can handle the expected workload. This can be done in a number of ways, like for example, increasing the world on the systems and have the system stressed to its limits. Systest.com (2001) define performance testing as testing to ensure that the system meets all performance requirements and to identify the limits of the system. The conditions under which a system should be test for performance are normal or nominal conditions, high load conditions, high volume conditions and stress conditions. It is testing conducted to evaluate the compliance of a system or components with performance requirements (ApTest, 2003).

- **Platform**

Describes a unique and complete computer system. Each platform, such as IBM, UNIX, or Macintosh, approach the solutions to computer-use problems from different perspectives

- **Reference time**

The amount of time that a particular benchmark took to run on a specific reference platform.

- **Regression Testing**

Once a product has been tested and fixes have been made to a product, the product needs to be tested again to ensure that all reported bugs have been fixed. Regression testing is a way to ensure that no new problems have occurred.

- **Response time**

The amount of time from when an action is requested until the time that the request completes and is returned to the requestor.

- **Server**

In LANs (local area networks), the server manages the traffic on the network, orchestrating demands on peripheral devices and central files so that multiple users' requests get responses in a timely and efficient manner.

- **Testbed**

The entire setup, including the System Under Test and any external system used to drive or coordinate or monitor the benchmark.

- **Unit Testing**

Unit testing is often the first step in testing software. Its intent is to build confidence in the correctness of a unit (Delamaro et al, 2001). It is some times called component testing.

- **Upload**

This refers to the transfer from the local machine to the remote one

- **Virtual User**

A virtual user is a simulation of a real user created by software. How many simultaneous virtual users one needs to simulate depends on a number of many factors, starting with how one expresses his Web site's capabilities.

- **Web site**

A set of pages that are meaningfully linked together

- **Workload**

The workload is he definition of the units of work that are to be performed during a benchmark run.

- **WWW**

An acronym for the World Wide Web. A hypertext-based distributed information system. It is also a hypermedia retrieval systems for information.

Appendix B: Description of the Automated Testing Tools Used in this Study

B1. Web Performance Agent

Performance Software (by TeamQuest Corporation) installs quickly and immediately begins collecting data on the entire environment. In no time it works as a powerful analysis tool to investigate problems and identify trends, automatically publishing reports to the Web for review, and viewing system health and alerts for all of the systems from a single console. One can go one step further and build analytic models of the system for what-if scenarios and prediction (TeamQuest, 2001).

B1.1 Key capabilities or features of this tool are:

- it monitors overall Web server performance including throughput, connections per second, errors per second, file transfer sizes, and request times;
- collects Web server performance data by File Types (images, sounds, video, dynamic, documents, compressed, bitmaps, etc)
- collects Web server performance File Transfer Size (to 4KB, 4 to 16KB, 16 to 64KB, 64 to 256KB, and over 256KB)
- collects Web server performance data by HTTP Request Type
- collects HTTP Reponses Code rates
- provides a list of the top files accessed at from the Web server including the filename, file type, access count, file size, and request times for each file.
- supports multiple "Web servers" on a single system
- allows one to correlate changes in Web server utilisation to its effects on overall system performance

B2 TestMaker (version 4.0)

It was originally known as Load. TestMaker (by PushToTest) offers free open-source that automates Web service systems for reliability, functionality, scalability and performance. TestMaker is a framework and utility to build and run intelligent test agents that implement user behaviours and drive the system as users would. It is licensed under a free Apache-style open source license.

B2.1 Key capabilities or features of this tool are:

- TestMaker features a graphical user interface, including a better execution window that will show a Stop icon to halt a running agent.
- currently, one needs to right-click the mouse button over the running agent's name and choose Terminate Process. This is not very friendly.
- it has a protocol handler for Java Management Extensions (to support SNMP for both as client and service).
- it has also a protocol handler for Peer-to-Peer) P2P and JAX-RPC and Apache Axis to enable testing from multiple SOAP stacks.
- log service – agents have the option to log to the output window or a log service. The log service can save log entries to a disk file or transmit them to another service.

B3. Web Performance Trainer (version 2.4)

Web performance Trainer (by WebPerfromance .Inc) can simulate hundreds of users using a second hitting a Web server at connection speeds 14.4kbps to 100mbps. It can help one find performance bottlenecks, increase performance, or do capacity planning. It can be used in finding out how many users a company's Web site can handle. It can be loaded and used within few minutes. It is based on recording browser/server interaction rather than emulating a browser, which increases its accuracy. While easy to use, it is also sophisticated. It uses the same multithreading technology found in expensive load testing tools, while giving the tester control to monitor and edit the data-stream. With Web performance Trainer one can find

performance bottlenecks early during the development or configuration process so they can be fixed before one's users can complain (Webperformance, 2002).

B3.1 Key capabilities or features of this tool are:

- supports all browsers
- supports all Web servers
- user simulation
- runs on Windows NT, Linux, Solaris, and most UNIX variants
- simulates users at variable connection speeds
- reports statistics at the transaction, Web page, and hit level
- no scripting required
- groups test cases into virtual load profiles to simulate existing or expected usage patterns
- tests complex Web pages containing links to multiple servers and ports
- each virtual user can have unique parameters for any HTTP parameters, including cookies, Form fields, etc.
- supports any kind of back-end process, including Active Server Pages (ASP), Applets, servlets, plugins, ActiveX Components, ISAPI, and cgi-bin
- records and allow viewing of the exact bytes flowing between the browser and server.
- graphs a variety of statistics, or reports data to a spreadsheet

These three tools were evaluated along several criteria and as a result of that was selected a tool for the experiments.

Appendix C: Automated Testing Tools Comparison Criteria

Comparison Criteria – For Test Tool Evaluation				
Criteria (c)		WebPerformance Trainer	TeamQuest	TestMaker–PushToTest
		Score (1-10)	*Score (1-10)*	*Score (1-10)*
C1	All users can be queued to execute a specified action at the same time	8	6	3
C2	Automatic generation of summary load testing analysis reports	10	10	5
C3	Ability to change recording of different protocols in the middle of load-recording session	--	--	--
C4	Actions in scripts can be iterated any specified number of times without programming or rerecording of the script	10	10	5
C5	Different modem connection speeds and browser types can be applied to a script without any rerecording	10	8	--

C6	Load runs and groups of users within load runs can be scheduled to execute at different times	6	4	8
C7	Automatic load scenario generation based on load testing goals: hits/seconds, number of concurrent users before specified performance degradation, and so on	8	5	5
C8	Cookies and sessions IDs automatically correlated during recording and playback for dynamically changing Web environments	--	--	--
C9	Allows for variable access methods in a single scenario: modem simulation or various line speed simulation	8	10	6
C10	Ability to have data-driven scripts that can use stored pool of data	5	5	10
C11	Allows for throttle control for dynamic load generation	10	10	10
C12	Allows for automatic service-level violation (boundary value) checks	5	4	8

C13	Allows for variable recording levels (network, Web, API, and so on)	8	6	6
C14	Allows for Web application server integration	--	--	--
C15	Supports workload, resource, and/or performance modelling	10	10	6
C16	Can run test on various hardware and software configurations	10	10	10
C17	Supports headless virtual user feature (Web, database etc)	10	10	8
C18	Scales to 500 – 1000 virtual users	10	8	5
C19	Simulated IP address for virtual users	10	5	7
C20	Thread-based virtual user simulation	--	--	--
C21	Process-based virtual user simulation	10	10	10

C22	Centralised load test controller	10	10	10
C23	Allows for reusing scripts from functional test suite	10	5	10
C24	Compatible with SSL recording	--	--	--
C25	Compatible with one or more of the relevant technologies: SAP, CORBA, COM	--	--	--
C26	Compatible with one or more of the relevant technologies: NT, Win2000, WinME, UNIX	10	10	10
C27	Monitors various tiers: Web server, database server, and app server separately	10	8	6
C28	Supports monitoring for on or more of WebLogic, Apache, MS SQL Server, IIS, Netscape Web Server	10	10	10

C29	Supports monitoring for one or more of the relevant technologies: NT, UNIX, XWindows, Win2000	10	10	8
C30	Monitors network segments	10	10	8
C31	Supports resource monitoring	10	10	8
C32	Synchronisation ability in order to determine locking, deadlock conditions, and concurrency control problems	10	--	--
C33	Ability to correlate any metrics from all monitors to identify performance bottlenecks	10	8	8
C34	Ability to provide client-to-server response times	10	8	4
C35	Ability to provide graphical results and export them to common formats	10	10	--
C36	Ability to provide performance measurements of data loading	10	10	10

C37	Requires scripting?	10	10	1
C38	Can do a "what-if analysis"	10	10	5
C39	Supports all browsers – browser independent	10	10	8
C40	Support from vendor available	10	10	5
C41	Ability to use Virtual Database Users	--	--	--
	Total Score ->	**308**	**270**	**223**

Other Comparison Criteria not Allocated with Scores				
C42	Cost of the tool	Offered free of charge if used for student research purposes		
C43	Potential problems			Has installation problems which stems from TestMaker not recognising the Java installation on ones computer. It also fails to recognise the space in 'Program Files' directory

Appendix D: Questionnaire

Section A: Request for Cooperation

Good day: Sir/Madam, Sawubona

My name is Mike J Mhlabane [BCom(Hons-IS)]. I am an MTech(IT) student at the Durban Institute of Technology. I would like to request your cooperation in assisting me with my research for my studies.

Purpose: to determine the extent to which online shoppers are happy (or unhappy) with their online shopping experiences on e-commerce Web sites. Please allow me approximately 10 minutes of your time.
Please answer the following questions to the best of your ability. You have the option to remain anonymous, so please feel free to express your opinion openly.
This research is intended to improve your experience as an online shopper in the e-commerce Web systems.

Thank you for your time and cooperation in completing this questionnaire.

Section B: Identification Data (Optional)

Your Name	
Your Address	
Email Address	

1. Please indicate your age

Under 20	
20 – 30	
30 – 40	
40 – 60	
Over 60	

2. **Are you male or female?**

Male	
Female	

Section C: Main Content

3. **Do you use the Internet?**

Yes	
No	

4. **Have you ever had any tuition/lesson on how to use the Internet?**

Yes	
No	

5. **How often do you use the Internet?**

Daily	
Weekly	
Monthly	
Occasionally	
Other – please specify:	

6. **Have you ever purchased a product or service on the Internet?**

Yes	
No	

7. **If you answered yes to the above question, how often do you purchase products or services on the Internet?**

Daily	
Weekly	
Monthly	
Occasionally	
Other – please specify:	

8. What inspired or stimulated you to shop online?

Interest	
Profession	
Convenience	
Other	

9. Do you buy online using Internet at:

Work	
Home	
School	
University	
Other – please specify:	

10. What data transmission method are you using to connect to the Internet?

Dial-up line	
ISDN (1-line – 64 kbps)	
ISDN (2-lines – 128 kbps)	
Local Area Network (LAN, Ethernet	
Satellite (Direct PC)	
ADSL (Asymmetric Digital Subscriber)	
ATM (Asynchronous Transfer Mode)	
Cable Modem	
Other (specify)	

11. Have you been dissatisfied with any e-commerce Web site's performance?

Yes	
No	

12. What was the nature of the problem/cause of dissatisfaction?

Disconnected connection	
Web page went down/stopped loading	
Web page loading very slowly	
Web page not clear on functionality	
Poor design of Web page/Not appealing to capture your attention	
Other – please specify:	

13. What do you do when you find that a Web site has unsatisfactory response time/performance?

Abandon the Site and never come back to it	
Abandon the Site and check it again later	
Send an email to the Web site master	
Seek alternative Web site that offer the same product or service	
Wait until Web page has fully loaded and do what you want to do	
Other - please specify:	

14. Are you aware of other (alternative) Web sites that offer the same product or service as that which you buy from your current Web site?

Yes	
No	

15. If yes (above) have you considered switching to the other Web site?

Yes	
No	

16. What would possibly motivate you to switch to the alternative Web site?

Poor performance	
Lack of required information	
Lack of Product/Service	
Other (please specify)	

17. Are you aware of the minimum response time your Web pages should take to load?

Yes	
No	

18. Indicate your expected response time of Web pages

24 seconds	
23 seconds	
18 seconds	
16 seconds	
30 seconds	
Other (specify)	

19. If the Web site's response time was fairly acceptable would you return to it?

Yes	
No	

20. If the Web site's response time was NOT acceptable would you return to it?

Yes	
No	

21. If yes (from above) what would make you return to the Web site?

Web site has good offers	
No other Web site providing the same service	
Not aware of alternative Web sites	
Other (please specify)	

22. Have you ever had a site suddenly going down/becoming unavailable while you were busy with your online shopping?

Yes	
No	

23. Do you know of any people who have had bad experiences with their online shopping?

Yes	
No	

Thank you for cooperation. It is highly appreciated.

Appendix E: The Test Results Collected During the Test Experiments

Table E1: Batch 1 - Test Results from the Test Experiments

	Time	Repeats	MinTFB	Avg TTFB	Max TTFB	Min Dur	Avg Dur	Max Dur	Users	Errors
1	2:24:00	39	0.00	0.00	0.00	3.43	3.44	3.44	20	0
2	0:20:20	45	0.00	0.00	0.00	3.44	3.44	3.44	20	0
3	0:20:30	46	0.00	0.00	0.00	3.44	3.44	3.44	22	0
4	0:20:40	46	0.00	0.00	0.00	0.00	0.00	0.00	24	0
5	0:20:50	48	0.00	0.00	0.00	3.44	3.44	3.46	24	0
6	0:21:00	50	0.00	0.00	0.00	3.42	3.42	3.42	27	0
7	0:21:10	52	0.00	0.00	0.00	3.44	3.44	3.46	27	0

Table E2: Batch 2 - Test Results from the Test Experiments

	Time	Repeats	MinTFB	Avg TTFB	Max TTFB	Min Dur	Avg Dur	Max Dur	Users	Errors
1	20:00:01	43	0.00	0.00	0.00	3.43	3.58	4.01	100	0
2	20:20:13	45	0.00	0.00	0.00	3.44	3.52	3.58	100	0
3	20:40:07	48	0.00	0.00	0.00	3.44	3.42	3.48	120	0
4	21:00:42	48	0.00	0.00	0.00	4.21	4.22	4.23	135	0
5	21:20:48	50	0.00	0.00	0.00	4.17	4.19	4.19	135	1
6	21:40:29	52	0.00	0.00	0.00	4.39	4.42	4.42	142	0
7	22:00:16	57	0.00	0.00	0.00	4.47	4.49	4.51	142	0

Table E3: Batch 3 - Test Results from the Test Experiments

	Time	Repeats	MinTFB	Avg TTFB	Max TTFB	Min Dur	Avg Dur	Max Dur	Users	Errors
1	0:00:00	87	0.00	0.00	0.00	6.39	7.01	7.11	200	4
2	0:20:13	92	0.00	0.00	0.00	6.57	7.06	7.13	250	3
3	0:30:17	89	0.00	0.00	0.00	7.21	7.38	7.52	280	5
4	0:40:33	95	0.00	0.00	0.00	8.01	8.09	8.09	300	3
5	1:00:03	103	0.00	0.00	0.00	9.13	9.28	9.32	320	8
6	1:20:11	116	0.00	0.00	0.00	10.42	10.49	10.52	350	11
7	1:40:29	129	0.00	0.00	0.08	11.04	11.07	11.07	400	17

Table E4: Batch 4 - Test Results from the Test Experiments

	Time	Repeats	MinTFB	Avg TTFB	Max TTFB	Min Dur	Avg Dur	Max Dur	Users	Errors
1	13.00	96	0.00	0.00	0.00	8.05	8.05	8.11	300	7
2	13.10	101	0.00	0.00	0.00	10.37	10.37	10.49	340	2
3	13.20	113	0.00	0.00	0.00	11.02	10.47	10.53	350	7
4	13.30	109	0.00	0.20	0.30	12.23	10.58	11.01	366	13
5	13.40	121	0.30	0.60	0.41	13.34	13.34	13.41	371	21
6	13.50	119	0.50	0.56	0.59	14.07	14.07	14.11	383	23
7	14.00	137	0.58	1.01	1.02	16.13	16.13	16.17	409	26

Table E5: Batch 5 - Test Results from the Test Experiments

	Time	Repeats	MinTFB	Avg TTFB	Max TTFB	Min Dur	Avg Dur	Max Dur	Users	Errors
1	16.30	94	0.00	0.00	0.00	8.05	8.05	8.11	300	7
2	16.40	99	0.00	0.00	0.00	10.37	10.37	10.49	340	2
3	16.50	116	0.00	0.00	0.00	10.47	10.47	10.53	350	7
4	17.00	111	0.00	0.20	0.30	10.59	11.00	11.01	366	13
5	17.10	118	0.20	0.55	0.41	13.34	13.34	13.41	371	21
6	17.20	122	1.00	1.05	1.02	14.07	14.07	14.11	383	23
7	17.30	142	1.09	1.13	1.07	16.13	16.13	16.17	409	26