TOWARDS A FRAMEWORK FOR DETERMINING A PLATFORM FOR TEACHING WEB APPLICATION DEVELOPMENT IN TERTIARY INSTITUTIONS IN SOUTH AFRICA

by

JOHNSON OLUMUYIWA DEHINBO

submitted in part fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in the subject

INFORMATION SYSTEMS

at the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR: PROF L M VENTER

JOINT SUPERVISOR: MRS A J GERBER

JULY 2006

Student number: 3174-123-1

DECLARATION

I declare that the dissertation titled "Towards a framework for determining a suitable platform for teaching Web application development in tertiary institutions in South Africa" is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.

…….…….. ……..

MR. J. O. DEHINBO DATE

TOWARDS A FRAMEWORK FOR DETERMINING A PLATFORM
FOR TEACHING WEB APPLICATION DEVELOPMENT
IN TERTIARY INSTITUTIONS IN SOUTH AFRICA

SUMMARY

This study develops and applies a conceptual framework that can be used to evaluate dynamic Web platforms in order to determine a platform for teaching Web application development in tertiary institutions. The framework is specific, yet comprehensive and supported with theory and empirical experimental programming results.

We first identify the concepts of Web application development and the constraints to be satisfied by a platform for teaching identified concepts. Then we establish various criteria that will enhance the teaching of the concepts. We also establish qualities and experiment that will ensure that a selected platform is easy to use, fast, portable and affordable.

A spreadsheet tool is developed to apply the framework and enable users to customize the framework by varying the criteria's weights. The developed framework is tested by the evaluation of the suitability of Java Servlets, JSP, ASP and PHP with PHP emerging as a suitable platform.

Key terms: Internet; Web; programming; languages; scripts; middleware; dynamic platform; framework; evaluation; usability.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ADO – ActiveX Data Objects

ASCII – American Standard Code for Information Interchange

ASP – Active Server Pages

API – Application Programs Interface

BASIC – Beginners All purpose Symbolic Instruction Code

C/C++ – A structured and object-oriented programming language

CFML – ColdFussion Markup Language

CGI – Common Gateway Interface

COBOL - Common Business Oriented Language

COM/DCOM – Component and Distributed Component based language

CORBA – Common Object Request Broker Architecture

DBASE – DataBase programming language

DBMS – Data Base Management Systems

DDL – Data Description Language

DML – Data Manipulation Language

EJB – Enterprise Java Beans

FastCGI – Fast Common Gateway Interface

FORTRAN – Formula Translator

FOXPRO – A Data Base programming language like DBASE

GUI – Graphical User Interface

HTML – Hypertext Markup Language

HTTP – HyperText Transmission Protocol

IDE – Interface Development Environment

IIS – Internet Information Service

ISP – Internet Service Provider

ITS – Integrated Tertiary System

J2EE – Java 2 Enterprise Edition

JavaScript – Java Scripting edition

JDBC – Simply means JDBC (not Java Data Base Connectivity as one might think)

JSDK – Java Servlet Development Kit

JSP – Java Server Pages

LISP – A logic programming language

LOC – Lines of Code

MSDOS – Microsoft Disk Operating System

MVC – Model View Controller

NNTP – Network News Transmission Protocol

ODBC – Ole DataBase Connectivity

OOP – Object-Oriented Protocol

ORACLE – A Data Base Management System & programming language

ORB – Object Request Broker

OS2 – Operating System 2

PASCAL – Programming Language designed by Pascal

PDA – Personal Digital Assistant

PHP – Personal Home Page Hypertext Processor

PWS – Personal Web Server

RDBMS – Relational DataBase Management Systems

RMI – Remote Method Invocation

RPC – Remote Procedure Call

RPG – Report Page Generator.

SIMULA – An early object-oriented language

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

TECLA – Technikon Computer Lecturer Association

UDDI – Universal Description, Discovery and Integration

URL – Uniform Resource Locator

WWW (W3C) – World Wide Web (World Wide Web Consortium)

WYSIWYG – What You See Is What You Get

VBScript – Visual Basic Scripting Edition

XML – Extended Markup Language

DEDICATION

Soli Deo Gloria

=====o=====

This is

dedicated to

to the Glory of the

Almigthy God

who gives us the

enablement

ACKNOWLEDGEMENTS

My special acknowledgement goes to the South African Institute for Race Relations (SAIRR/TELP) for sponsoring my tuition fees for the course modules for 3 years between 2000 to 2002 and the Tshwane University of Technology (formally Technikon Northern Gauteng - TNG) for paying my tuition for the remaining 4 years (2003 to 2006). Special thanks also go to the TNG Central Research Committee as well as to the Research Capacity Building project (by the Netherlands government) for sponsoring various components of the research study.

I solidly acknowledge the sense of direction given by my supervisor, Prof L.M. Venter, and my co-supervisor, Mrs. A.J. Gerber, both in personal discussions and in electronic conversations. Due to them, I can confidently say I have learnt a lot, from which my students are also benefiting.

My special thanks go to my colleagues at the Tshwane University of Technology especially Mr M.M. Madubanya, and to others including Mr. Yemi Odunaike, Mr. Manoj Lall, Mr. Bellington Makoba, Mr. Sam Akinsola, Mr. Mike Adeyemo and some others. I am also grateful to Dr. Kok, the Research Director, and to Prof. M. Herselman, Chairperson of the ICT Faculty Research Committee for encouraging us on research, publications and presentations. Thanks also goes to Mrs Maria Sebate, my junior brother, Dr. Bank Dehinbo, and my in-law, Mr Muyiwa Oluokun, whose assistances in various forms give me time to concentrate on the project. Moreover, I am grateful to Pastor Clement Ibe for constantly reminding us that "we are able...", and giving us some success mentality tips, one of which is tenacity.

Last, but not the least, I thank the two most important women in my life, my darling wife, Mrs Kehinde Dehinbo, and my mother Mrs Rachael Dehinbo for handling various other important life projects thereby ensuring that the period of this study doesn't imply an "impasse" in my other life responsibilities. Most importantly, I thank the Almighty God for seeing me through.

ABSTRACT

In order to determine a platform for teaching Web application development in tertiary institutions, this study develops and applies a conceptual framework that can be used to evaluate dynamic Web platforms. The framework is supported with evidence from relevant theoretical references, previous studies, experience and empirical experimental programming results. The framework is both specific and comprehensive unlike previous studies which have either simply compared platforms without an established specific yet comprehensive framework or have compared them in terms of one specific factor only.

In developing the framework, we first identify the concepts of Web application development that should be taught, and the constraints to be satisfied by a platform for teaching identified concepts. Then we establish various criteria that will enhance the teaching of the identified concepts, as well as qualities that will satisfy the constraints to ensure that a selected platform is easy to use, fast, portable and affordable for second year students. A latency experiment to estimate the speed of processing using various platforms is further incorporated into the framework.

A spreadsheet tool is developed to apply the framework by automating the calculation of scores for the criteria and to enable users to customize the framework by varying weights allocated to the various criteria. The developed framework is tested by the evaluation of the suitability of Java Servlets, JSP, ASP and PHP for teaching at the Tshwane University of Technology, with PHP emerging as a suitable platform.

Key terms: Internet; Web; programming; languages; scripts; middleware; dynamic platform; framework; evaluation; usability.

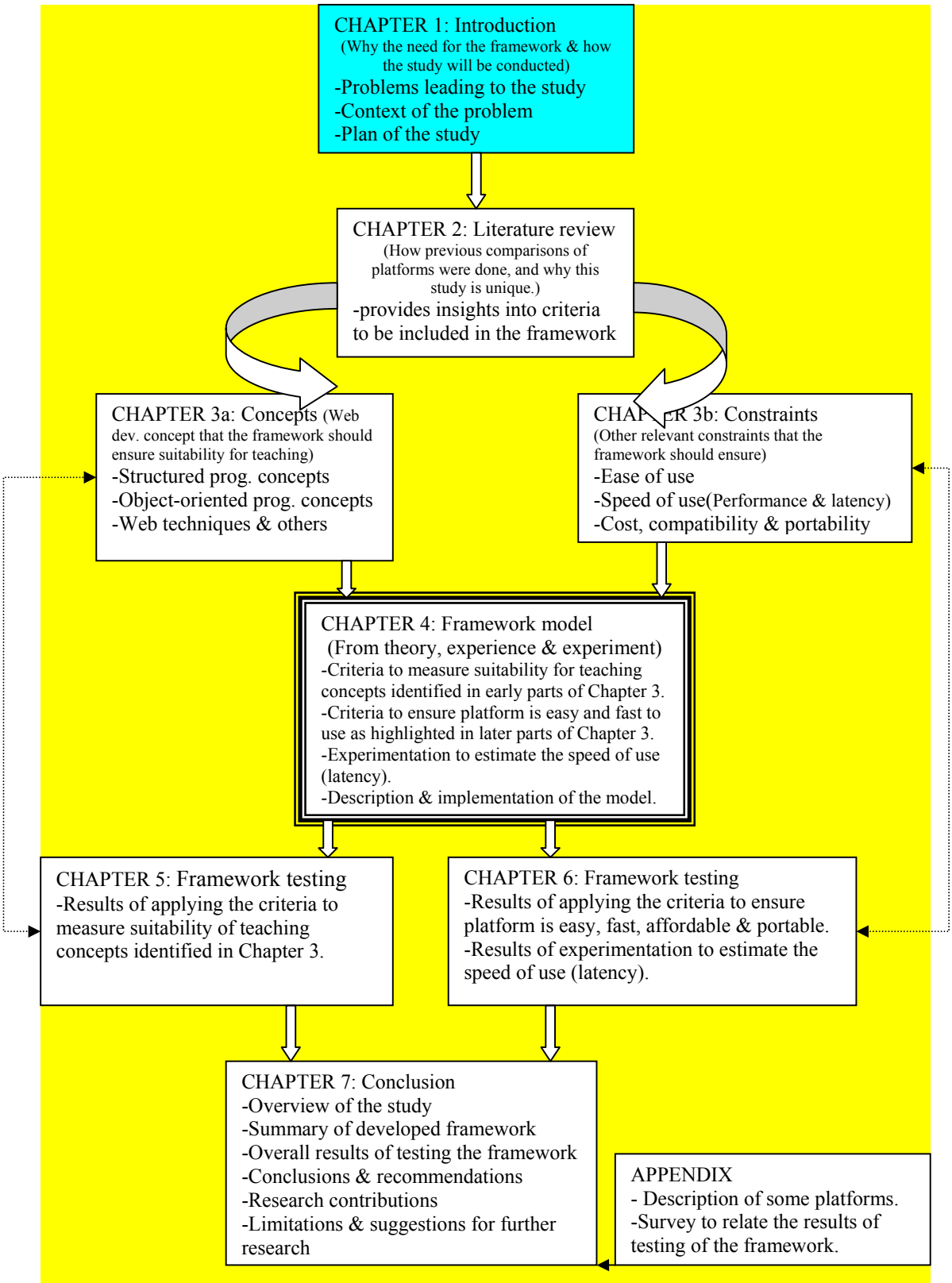

Figure 1-0. Dissertation map highlighting the relativity of chapter 1 to the rest of the dissertation.

CHAPTER 1

1 BACKGROUND OF THE STUDY

1.1 Introduction

The overall goal of this study is to develop and apply a conceptual framework that could be used to determine a platform for teaching Web application development, as part of the curriculum of the Web Management specialization of the National Diploma in Information Technology at tertiary institutions in South Africa. The dissertation map in figure 1.0 illustrates the overall outline and structure of the dissertation.

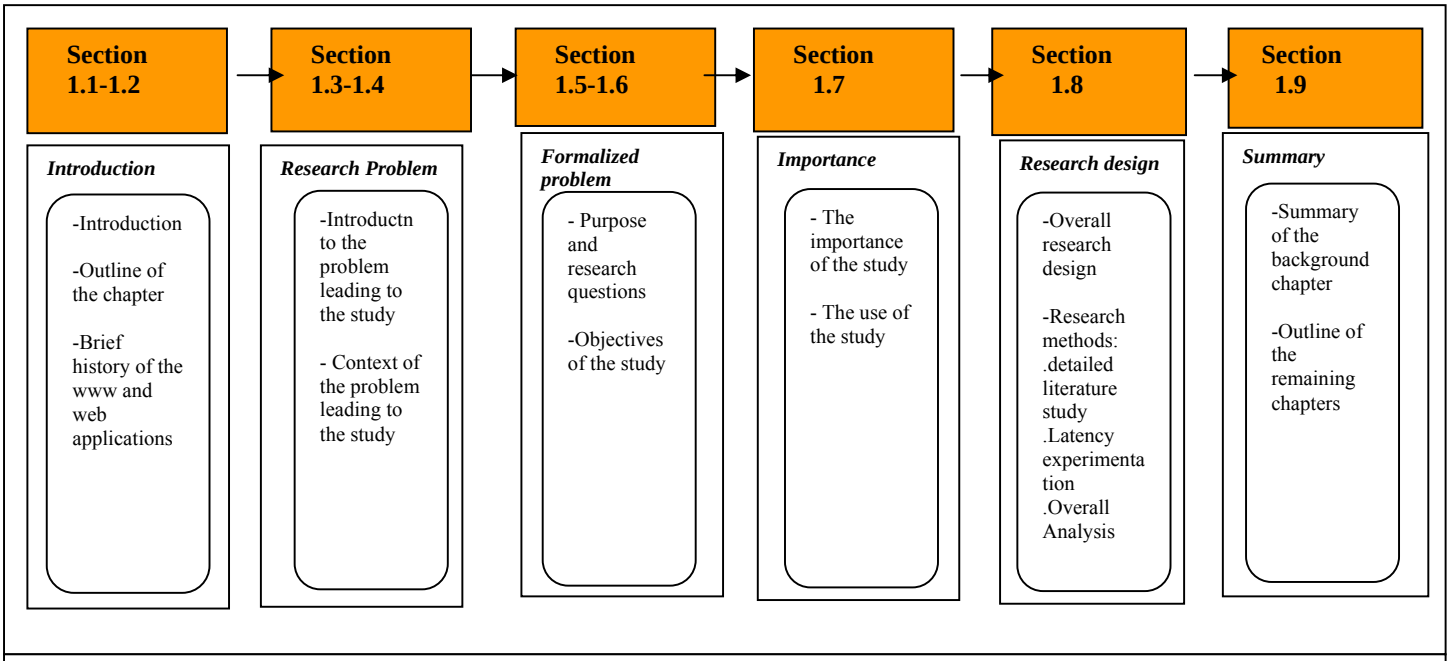

Figure 1-1. Pictorial outline of chapter 1.

The first step in solving a problem is the identification of the problem which is presented in chapter 1. Figure 1.1 above gives a pictorial outline for chapter 1. Section 1.1 to section 1.4 introduces the problems leading to the study while section 1.5 gives the purpose and research questions. This is followed by the objectives and the importance of

the study in section 1.6 and section 1.7 respectively. Section 1.8 presents the research design and methodology while section 1.9 summarizes chapter 1 and outlines the remainder of the dissertation.

In order to for the framework to determine a suitable platform, it would involve the fulfillment of certain general requirements that are considered important to the effective comprehension and future usefulness of Web application concepts. These requirements described below, would later be refined into appropriate criteria substantiated from literature:

Effective teaching of dynamic Web applications development: This requirement implies that the selected platform will:

- be suitable for the purpose of teaching Web applications development to second year students of tertiary institutions;
- utilize the program design principles as well as the knowledge of programming languages taught to illustrate those principles;
- be easy to use by second year students of the Web applications development specialization of the National Diploma in Information Technology;
- be suitable for real life programming that students will be exposed to when they graduate;
- integrate with other relevant components of the curriculum of the Web applications development specialization of the National Diploma in Information Technology;
- be affordable to allow for individual ownership of the platforms by students, so that they can practice at home;
- have commands with syntax, semantics and meanings that will enhance comprehension; and
- have a reasonable number of unique basic syntactical components that avoid redundancy in some type names and commands.

Allowing an efficient teaching environment: The constraints imply that the framework will allow for an efficient teaching environment that does not distract students from their purpose, as the selected dynamic Web platform will:

- be suitable for completing tasks in a reasonable time;
- offer features that promote readability of the code;
- be equipped with tracing and debugging help facilities;
- offer a useful and smartly integrated editing environment;
- be equipped with context sensitive "helps";
- have more higher-level commands that accomplish tasks with few statements;
- be portable among major operating systems;
- have high throughput by compiling and executing quickly thereby ensuring suitability for the recurrent testing and debugging in a practical laboratory;
- use minimum disk space and memory resources; and
- have simpler architecture and execution steps to facilitate speedy operations.

The development of the framework will therefore have various criteria against which we will measure different platforms, and then, based on these measurements, we will select the best platform for teaching Web application development in tertiary institutions. This will contribute towards ensuring the development of well-equipped graduates specializing in Web application development.

The purpose of the application of the framework is not to persuade users that one platform is better than another, but to rather assist users to make a more informed decision on which platform is more suitable for teaching Web application development in higher institutions. This implies that the framework will ensure that, although the chosen platform may or may not be the most powerful or most common on the market, it will be one that has desirable features that contribute to the attainment of Web application development skills by second year students of the National Diploma in Information Technology.

In order to appreciate the importance of such a framework, we need to appreciate the problems leading to the study. These commenced with the wide use of the World Wide Web, which led to the multitude of platforms currently available for developing Web applications.

1.2 Brief history of the World Wide Web and Web applications

According to Schneider and Perry (1999, p.5), the Internet is a large collection of computers all over the world that are connected to one another in various ways, while the World Wide Web (WWW) is a subset of these computers, connected in such a way that makes them and their contents easily accessible to all computers in that subset. The origin of the Internet dates back to three decades ago, but the introduction of World Wide Web is a relatively recent event started in 1990 by Tim Berners-Lee (Deitel *et al.*, 2001, p.4; Schneider & Perry, 1999, p.5).

The importance of the Internet and the World Wide Web is discussed by Deitel *et al.* (2001, p.12), who states that the Internet will surely be listed among the most important and profound creations of mankind, and that today's applications can be written to enable communication among the world's hundreds of millions of computers. According to Yerkey (2001), the attitude of some users has become "if it is not on the Web, it does not exist". While this statement might not be generally true, it does seem to set a trend.

How does the Internet achieve that much importance? Deitel *et al.* (2001, p.12) further explains that the Internet mixes computing and communication technologies, while Fraternali (1999) states that the open architecture, multi-medial and browsing nature of the Web facilitate the integration of different types of content and systems. These combined technologies have a profound effect on today's transactions, as confirmed by Mackinnon and Denne (2001, p.6), who claim that with access to Internet, unlike using desktop applications alone, one can perform a huge variety of actions, from games, gossip, messages, music to academic research and shopping.

The Internet and the World Wide Web have made Web applications more relevant today. Mercer (2003, p.4) notes that while desktop application programs have been useful for many years, the rise of the Internet and the World Wide Web has made online or Web applications more appealing for business transactions worldwide. This, he maintains, is due to the fact that unlike desktop applications, Web applications can coordinate the input of many users in real time and can connect to other data sources in real time.

Moreover, customers need access to products and services on a 24-hour, 7 days a week basis. Deitel *et al.* (2001, p.5) state that the easiest way to provide such support is to move operations online. As a result, traditional "brick and mortar" stores are constantly being replaced by a multitude of electronic storefronts which populate the World Wide Web. A good example of this is the banking system in which most operations are being moved online as it becomes clear that the Web, unconstrained by geographic boundaries, is a more efficient vehicle for their services and allows them to work on a truly global scale (Deitel *et al.*, 2001, p.5).

In educational institutions, we observe that important applications, such as the Integrated Tertiary System (ITS) used for tasks such as registration, time tabling and academic records systems are now implemented on the Web, so that authorized users can log in remotely. Even the email systems are now Web-based. This is an improvement over the situation that existed when a worker has to suspend all work-related activities while being away from the office computer.

However, the above-mentioned improvement did not happen in a flash. The next section discusses the progression from the use of Hypertext Markup Language (HTML) in static Web pages to dynamic Web pages in Web applications using dynamic Web platforms.

1.2.1 Static Web Pages and Dynamic Web pages

According to Apte *et al.* (2003, p.2), the World Wide Web first emerged as a medium to render hypertext documents, stored on the Internet, onto a user's computer, using special software (the browser) and the Hyper Text Transmission Protocol (HTTP). This involves

the use of the Hypertext Markup Language (HTML). Pages created this way are said to be static because the content does not change until the page is modified (Yerkey, 2001).

In order to easily create Web pages to display static data, systems such as Microsoft Word include a "Save As HTML" feature. When a browser requests such an HTML file, the same content is displayed, until the HTML file is changed or replaced with another (Horstmann, 2002, p.1003). According to Yerkey (2001), pages created in this way are a "snapshot in time"; they do not change as the content changes. Yerkey (Lam, 1997) confirms that "publishing and maintaining a large number of static Web pages result in a maintenance nightmare". Worse, users cannot search the database for items of data; they can only see what was previously saved in HTML format (Yerkey, 2001).

Today the use of the Web has undergone tremendous progression, from the early serving of static Web pages to various dynamic Web applications. In this context, we refer to a Web application as a program developed to run on a Web page in order to transform a static Web page into something more dynamic that allows interaction, computation and remote access to up-to-date information.

Rather than modifying the Web page to obtain a recent view of the contents, the data values for the content are stored on databases. These database can be independently modified to reflect current values and users can interact via browsers with programs that access the up-to-date information from the database. The interaction and the up-to-date variability of information displayed make the system dynamic (Apte *et al.*, 2003, p.2).

Dynamic Web applications allow the serving of up-to-date information, accessible interactively from a database (Apte *et al.*, 2003, p.1; Horstmann, 2002, p.1003). Apte *et al.* (2003) note that the transformation from a static content distribution medium to an interactive dynamic medium is illustrated by the present wide use of the Web as the presentation layer for a host of on-line services such as email, shopping, banking and stock trading. Apte *et al.* (2003) notes further that the dynamic generation increases flexibility in customizing page content.

There are other reasons for the move from static Web pages to dynamic Web applications. Yerkey (2001) notes, too, that part of the reason for the slowness in using the Web for dynamic data access is that Hypertext Markup Language (HTML) has little processing power. Therefore, Web pages consisting of only HTML cannot search databases, make decisions based on conditions, do mathematical calculations, create animations, customize output for different users, respond to information entered on forms, and so on (Yerkey, 2001).

However, it is more interesting for Web users if websites allow interaction such that users can react to information on the websites. Such websites will serve dynamic content (Horstmann, 2002, p.1003) transformed into an interactive dynamic medium, where content on the Web is often personalized, interactive and therefore dynamically generated (Apte *et al.*, 2003, p.2). Davidson (2001) listed the benefits of dynamic websites. These include centralized data source, dynamic data, manageability, easier administration, content customization, streamlining and simplification of the process.

To search and to display dynamic content on websites requires the use of other languages, platforms or middleware systems (Yerkey, 2001). Mercer (2003, p.1) illustrates this clearly by pointing out that as a response to the need for easier integration of markup language code with server-side processing using server-based scripting languages, Microsoft created the Active Server Pages (ASP) technology. The name reflects how the technology makes Web pages “active” via programs running on the server. Like ASP, various platforms have emerged for generating dynamic Web content in Web pages by developing Web applications.

Given that the various platforms generate dynamic contents in Web applications, we would often refer to them as dynamic Web platforms where necessary, but most often, we would simply use the term “platforms”. Some of the newly emerged platforms are being used for teaching Web application development in tertiary institutions. To provide a background on these platforms, we examine the use of dynamic Web platforms in server-side processing.

1.2.2 Dynamic Web Platforms for server-side processing

Dynamic Web content is generated by a combination of a Web server, a dynamic content generator, and a back-end database not directly accessible from the client browser. The Web server serves all static contents and the dynamic content generator executes the code that captures the business logic of the Web application (Cecchet *et al.*, 2003, p.2). The dynamic content generator formats and assembles the results which are combined with the static content by the Web server to form an HTML output file. This is then forwarded as an HTTP response to the client browser. The browser formats and displays the page according to the HTML tags contained (Cecchet *et al.*, 2003, p.2).

According to Yerkey (2001), dynamic Web content generation may involve client-side processing and / or server-side processing. In client-side processing, the server sends a Web page containing both code and data back to the client (user's computer), and the user's computer does all the work. In server-side processing, the server calls up programs residing on the server. These programs process the data, and send the processed data back to the user's browser for display. The Web platform handles server-side processing, which can be accomplished using various platforms including Java, Python, Perl, ColdFussion, *etc.* The next section discusses how these led to the research problem.

1.3 Introduction to the problem leading to the study

The last section discussed the introduction of some Web platforms, the details of which are provided in Appendix 1. Vinoski (2004) indicates that the many platforms and vendors from which one can choose does not make choosing the one that will best solve one's problem an easy task. Which of these platforms will be most suitable for teaching Web application development?

This is important in view of Sebesta's (1996, p.2) statement that the depth at which we can think is influenced by the expressive power of the language in which we can communicate our thoughts. Sebesta (1996, p.3) further illustrates that the language in which programmers develop software places limits on the kinds of control structures, data structures and abstractions they can use, as well as on the form of algorithms they

can construct. This is relevant to dynamic Web platforms given that they involve the use of programming language constructs.

It might be plausible that Sebesta's assertions could be one of the reasons why there are new programming languages and platforms evolving day after day to overcome the limitations of some previously available platforms. Similarly, Bishop and Hurter (1999) mentions Trott (1997) stating that, "as time progresses, we see more and more new programming languages of all kinds appearing on the market." According to Bishop and Hurter (1999), some are developed by a tremendous work force, while others spring from a single person's work, but all aim to provide either new or improved functionality.

Web applications development is a subset of the general software development. The development process of either type of application is quite similar (Mercer, 2003, p.5) and follows similar trends. Lim (2002, p.2) confirms this by stating that the computing world is now swamped with Web technologies. We have enumerated some of these platforms in section 1.2.2. But why is that a problem? The emergence of new dynamic Web platforms one after another confirms the limitations to some of the platforms.

The various platforms will probably portray varying degrees of ease in learning and using them. If the dynamic Web platform used does not have the necessary qualities and features it will not be a good choice to use to teach the Web application development concepts, and will reduce the quality of designed systems. It will also result in students taking longer to develop their Web applications, and time allocated for laboratory sessions is limited. The next section examines the situation in more detail.

1.4 The context of the problem leading to the study

Lim (2002) states that information systems / computer science departments must reexamine their curricula in order to prepare students to face the challenge of being productive in a computing world that is now swamped with Web technologies. But how do we determine which platform will be most suitable for teaching Web application development to second year students in tertiary institutions? The Computer Studies

department of the old Technikon Northern Gauteng (TNG) (now the Soshanguve campus of the Tshwane University of Technology - TUT) has been confronted by this question.

The introduction of the new Diploma in Information Technology at the Technikon Northern Gauteng in the year 2001 motivated this study. As agreed upon at the Technikon Lecturers' Association in South Africa (TECLA) conference in Durban in 2000, the new diploma in Information Technology has nine specializations, including Web application development. The author is being charged with the responsibility of leading the way with regard to the design of the curriculum of the Web application development specialization.

The Web application development specialization teaches, among other things, how to develop Web applications. One important task, therefore, is the determination of the platform that will be used to teach the applicable concepts. Up to now the department has used various programming and scripting languages. These led to elements of repetition and confusion being reported by students. Some students resorted to absenteeism, stating that they had learnt the current lessons as they have been taught using other programming languages. However, they usually ended up failing such subjects.

Coupled with the above is the poor performance of students in C++ as a subject in the former Technikon Northern Gauteng in the last few years. In this case, the management (senate) of the former Technikon periodically requires explanations from the lecturers of why the "pass rate" is lower than 50% and suggestions of possible remedial action.

The above led to the study entitled "Determining a suitable programming language for the new B.Tech. programme in Information Technology at the Technikon Northern Gauteng (TNG)" (Dehinbo, 2004b). The study established the need to identify a suitable programming language that would be relatively easy for beginners to write useful programs, to incorporate other current and relevant IT technologies, and to offer some prospects for future industrial use and research.

As a follow-up to the recommendations of Dehinbo (2004b), there was a need to choose dynamic Web platforms that would complement the programming languages taught in the first year. It is important to bear in mind that our students at the University of Technology are mostly from a rural background and have no prior knowledge of computing and programming. Teaching them using a relatively complex platform may lead to a poor pass rate and low self-esteem and, again, the lecturers would need to account to the senate of the Institution for this low pass rate.

The second problem relates to the possibility of low levels of comprehension of Web application development students. Especially for novice or beginning programmers, comprehension is very important (Lahtinen, Ala-Mutka & Jarvinen, 2005; Wiedenbeck, 1999). Low levels of comprehension could be the case, if it takes students a long time to comprehend the use of the dynamic Web platforms being used[1].

The third problem originated from the slow speed of execution, due to the performance of dynamic Web platforms used by students. Why is the slow speed of execution a problem? On the surface, it may seem that the above problem results in a few seconds delay per typical execution. However, taking scalability into consideration, we consider an experiment in a previous study. Dehinbo (2005a) shows that for that experimental specification, as the number of records grows to up to 8000, the difference in the latency of the same program using different platforms becomes significant. This is up to the point that some platforms (ASP and JSP) were unable to process the records.

For a lower computer specification, the delay could only become significant when processing lower number of records, say 2000 records. Yet, the delay is still a problem. This is because delayed processing wastes part of the limited practical laboratory sessions. Moreover, if a student is testing a program to process such a database and waits few minutes to restart the computer because of the inability of the platform to process it, then valuable time will be wasted out of that 80-minute laboratory session.

[1] As an illustration, teaching will be faster with a platform that uses a one-line command to input data than one that nests three statements just to input data. To put this in another way, a prospective driver could drive better using an automatic car rather than a manual car.

The above suggests, as indicated by Marshak and Levy (2003) that measuring the delay experienced in using a Web application is of great importance. There are many other factors that could influence the delay, and such analysis is beyond the scope of this study. However, for every specific laboratory set-up, students will become frustrated if their developed Web application is slow. This becomes more important when students need to test and debug their programs repeatedly in the laboratory, where time is limited.

We may then ask: Why not just use faster networks or hardware? Firstly, upgrading the network is a university-wide task, and not a departmental task. Secondly, upgrading about 50 computers could be made difficult by budget constraints. Thirdly, as mentioned in the introduction, one of the requirements that our framework should ensure is that a platform that will work well on the average computer affordable by students would be preferable. Adding more memory may help, but that is an extra expense especially when the same computer executes programs in other platforms without delay.

In addition, there is a possibility that if a platform took so long when processing 8000 records up to the point that the computer has to be restarted, then if we upgrade the computer or the memory, it may possibly do same when processing, say, 15000 records or more[2]. In the words of Ashenfelter (1999, p.129), "processor upgrades don't make the application less processor efficient; they just make the problem less noticeable".

In summarizing the identified problems, the first is a need to identify a suitable platform for teaching Web application development. Secondly, a platform that is difficult to comprehend and utilize may have an adverse effect on the rate of mastery of the Web application development skills by students. Thirdly, there is the possibility of delay as latency increases with scalability while using some platforms. These problems demanded a systematic evaluation of some selected platforms in order to determine a suitable one for teaching Web application development in tertiary institutions.

[2] As an illustration, all things being equal, if a driver can only speed up to 60km/h despite the possibility of getting up to a maximum of 120km/h on a 1.3 engine capacity car, if given a 3.0 engine capacity car it is not likely that, with a maximum speed of 260km/h, he/she will cruise at a speed of 240km/h.

Unfortunately, analyzing dynamic Web platforms as well as programming languages with the aim of choosing a suitable one is not easy. Prechelt (2000) indicates that when it comes to the advantages and disadvantages of various programming languages, programmers and computer scientists alike usually hold strong opinions. However, analyzing programming languages, paradigms, development platforms and tools is still very important, though difficult, as illustrated below:

> Comparisons across programming styles, or paradigms, are difficult to carry out, but are nevertheless important for understanding how different styles of programming affect the learning of novice programmers. (Wiedenbeck *et al.*, 1999)

Moreover, Apte *et al.* (2003) note that a study of existing literature showed varying conclusions about the superiority of one dynamic Web platform over another. These arguments strengthen the need for detailed research so as to make appropriate choices backed by scientific facts. Apte *et al.* (2003) confirm the need for various factors by indicating that the decision on which dynamic Web platform should be employed in creating a Web application be based on factors including ease of programming, richness of features, maintainability, reliability and performance.

The eventual question then is: "How do we evaluate the Web platforms in order to determine the most suitable for teaching Web application development in tertiary institutions?" This led to this study of developing a framework for determining the most suitable platform for teaching Web application development to second year undergraduates in tertiary institutions. The framework needs to be supported with evidence from relevant literature, information from practising Web developers and empirical experimental programming results.

1.5 Purpose and research questions

In line with the above, the principal aim of this study is thus to develop and apply a conceptual framework for determining a suitable platform for teaching Web application development in tertiary institutions. The overall research question is:

How do we scientifically develop a framework to determine a suitable dynamic Web platform for teaching Web application development in tertiary institutions?

With this in mind, the study seeks to address the following research questions:

1. What are the concepts of Web application development that need to be taught to the second year students of Web application development?
2. How do we determine that a platform is suitable for teaching the identified Web application development concepts in our framework?
3. How would our framework ensure that a platform is easy to use and easy to learn for the second year students of Web application development?
4. How would our framework ensure that a platform executes fast by having low latency?
5. How would our framework ensure that a platform and its related components is affordable to students?
6. How would our framework ensure that a platform is portable across major operating systems?
7. How do we provide a way of varying the degree of importance of the various components of the framework?
8. How do we test our framework in determining a suitable platform among various platforms for teaching Web application development?

1.6 Objectives of the study

In order to answer the research questions, objectives must be formulated in clear, measurable and achievable/manageable terms. The objectives of the study are therefore given below:

1. Investigate related frameworks that compare similar programming languages and dynamic Web platforms.
2. Conduct educative study on the concepts of Web application development so as to gain insights into the suitability of dynamic Web platform for teaching the identified concepts.
3. Establish criteria to investigate the suitability of the dynamic Web platforms for teaching the identified concepts.
4. Establish criteria for investigating the *ease of use* and *ease of learning* of the dynamic Web platforms.
5. Establish criteria for investigating other constraints that the framework should satisfy, such as cost, architectural neutrality and portability among some popular operating systems and programming environments.
6. Establish criteria for determining the performance or latency of the designed Web applications in the remote retrieval of records from databases. This serves to estimate the throughput of the platform in the recurrent testing (compilation and execution) in a practical class.
7. Develop a spreadsheet tool to allow for the variability of the relative importance of the various criteria of the framework.
9. Test the framework by choosing a platform for teaching Web application development concepts, with the developed framework containing the various criteria.

1.7 Importance and use of the study

The main benefit of the study will be its contribution to the body of knowledge in terms of a framework developed for choosing the most suitable platform for teaching Web application development. This might enhance teaching of learners, increasing their potential in such a way that would lead in future to higher productivity in the software development and services industries. It will therefore be of benefit to institutions when deciding which dynamic Web platform to introduce for teaching Web applications development.

Therefore, the primary beneficiaries will be institutions that teach Web application development. In addition, software organizations, Web applications developers, researchers, students and individuals who utilize dynamic Web platforms in the development of their Web applications might use the framework and adopt it for their specific need.

1.8 Research design and methodology

This section presents the blueprint or the plan of how we are going to find answers to the research problems mentioned in section 1.5. This study uses a combination of different methods including non-experimental and experimental methods to obtain the data. The research design is an interpretive research approach in which we develop a framework from theory and experience. This will involve elements of descriptive studies and document analysis.

McMillan and Schumacher (2001, p.33) state that descriptive studies are non-experimental modes of enquiry in quantitative techniques which describes a system with the aim of characterizing it by using numbers. In this study, we use a descriptive approach to characterize the criteria developed for the framework, in order to use them to evaluate the platforms.

The measurement of the suitability of the platforms will involve the use of document analysis and descriptive modes of inquiry to characterize their features. There is thus a need to search the literature and other established bodies of knowledge to determine the features of the dynamic Web platforms so as to characterize and compare them according to some criteria that are paramount to their suitability.

In testing the framework, the ease of use of the dynamic Web platforms will sometimes be physically examined using observation as a research method. Bowling (2002, p.358) describes observation as a research method in which the investigator systematically watches, listens to and records the phenomenon of interest. In this study, the use of observation will be limited to an inquiry into the features and facilities available on the

selected installed platforms and their corresponding IDEs. Investigations will be conducted to determine the availability of facilities to aid the Web application development tasks.

The criteria for ensuring that a platform executes fast by having low latency will be supplemented with an experiment to determine the performance or latency of dynamic Web applications. McMillan and Schumacher (2001, p.32) state that an experimental mode of inquiry, the researcher manipulates what the subjects will experience. In the latency experiment part of this study, an algorithm will be manipulated by implementing it using the syntax of the various platforms.

The development of a tool for representing and manipulating the framework involves the use of modeling. According to Bowling (2002, p.141), models are abstract representations of the essential characteristics of phenomena of interest, thereby making explicit, the relationships and or comparisons between the characteristics. The form of modeling used in this study will consist of a set of criteria established to measure the suitability of the platforms. The development and implementation of a spreadsheet tool using Microsoft Excel allows for the variability of the degree of importance of the various criteria.

Finally, in line with the approach of Dix *et al.* (1998, p.431), who state that the best way to find out how a system meets users' requirements is to “ask the user”, an informal survey will be included in the Appendix and its results compared to the results obtained. The survey will provide feedback from the students who have been taught using the various platforms regarding their opinions on the perceived usefulness of the platforms.

The above justifies the use of the experimental and non-experimental method. The next section expands on these methods. Further details of these methods as they are applied are provided in chapter four and chapter five.

1.8.1 Building the framework's criteria

Using the concepts of Web application development that will be developed in chapter 2, we will establish various criteria that will be used to measure the suitability of the platforms for teaching the identified concepts. Other criteria will be established to measure the ease of use of the platform by second year students, as well as to ensure the low latency and high speed of execution of programs required for the recurrent testing and debugging that takes place in student practical sessions. Further criteria will be established to measure the affordability and portability of platforms.

At the lowest subdivision of the criteria, there is a series of questions that are used to evaluate the platforms. Users can choose to exclude or include the question respectively. This will be accomplished later by the use of applicable weights.

Using these criteria, we will seek answers to the questions and the availability of necessary features. These answers will be found in established texts, journals and authoritative websites which include those written by the designers of the platforms. The textbooks will be augmented by authoritative websites which include those of Sun Microsystems (designers of the Java platforms), Microsoft Corporation (designers of the ASP platform) as well as PHP websites and various sites for the applicable Web servers, such as Microsoft's Internet Information Service (IIS) and Tomcat.

1.8.1.1 Measuring scale for the criteria

The measuring of attributes in quantitative terms is very important in enhancing the understanding of our environment (Dhyani *et al*., 2002). Lord Kelvin (in Dhyani *et al*., 2002) summarizes this by stating that when one can measure what one is speaking about, and express it in numbers, one has demonstrated tangible knowledge about it. Therefore, using close-ended "Yes/No" questions, the measuring tool will have values on a scale of 1 to 3, where:

3 = "Yes",

2 = "Not quite or with some workaround", and

1= "No".

We have used a scale of 1 to 3 to avoid subjective situations where it will be difficult to distinguish between, say, a score of 3 or 4 in a scale of 1 to 5. The use of the 1 to 3 scale reduces the answer to whether the facility is available or not, or in between.

1.8.1.2 Establishing reliability and validity of the criteria development

It is important that a measuring scale or instrument be consistent and reliable. It should produce more or less the same accurate results every time it is applied, even when applied by different persons (Coertze & Heath, 1997, p.78). Also, Coertze and Heath (1997, p.79) indicates that validity is concerned with soundness or the effectiveness of the measuring instrument. Validity raises questions such as: Does the test measure what it is supposed to measure? How well? How comprehensively? How accurately does it measure it? (Coertze & Heath, 1997, p.79).

As a way of increasing validity, answers to the criteria questions will be sought from established and recognized textbooks (including text written by designers of the platforms), authoritative websites, and journal articles. We provide the accompanying references so that interested readers can verify or seek more information. This is supplemented with practical experiences and program tests confirming the satisfaction of some of the criteria established.

Also, the quantitative characterization and evaluation using numbers will enhance clarity in the choice of platform with the highest score. This is unlike just using qualitative sentences to evaluate the platforms, at the end of which it is difficult to say which platform is really more suitable.

Furthermore, the range of values applicable is between 1 and 3 instead of say between 1 and 5. This is to avoid subjective situations where it will be difficult to distinguish between, say, a score of 3 or 4 in a scale of 1 to 5.

1.8.2 Experimental approach to the latency estimation

An experimental approach will be used for the latency experiment. This will estimate the execution speed for programs written using each of the applicable syntax and system for each of the dynamic Web platforms. This is necessary in order to incorporate information on the suitability of the platforms for the recurrent program testing that takes place in a practical laboratory session into the framework.

In the experimental approach, a hybrid 3-tier client-server-database Web application will be developed for each of the selected platforms. The browser (client) interacts with the server program which accesses the database. The hybrid nature implies that the database resides on the server rather than on separate application server. Full details of this experiment are presented in Dehinbo (2005). Users of the framework can, however, set up their own latency experiment.

The experiment measures the latency of the designed Web applications using remote retrieval of records from database. This serves to estimate the throughput of the platform in the recurrent testing (compilation / interpretation - execution) in a practical class.

1.8.2.1 Establishing reliability of the latency experiment

As observed by Lerdorf and Tatroe (2002, p.311), if we execute Web applications and reload them several times, the time taken may fluctuate, affecting the reliability of the experiment. The danger in timing a single run of a piece of code is that one may not get a representative machine load – the server might be paging as a user starts, or it may have removed the source file from its cache (Lerdorf & Tatroe, 2002, p.311). Reliability can be increased by following the approach presented by Lerdorf and Tatroe (2002, p.311), which states that the best way to ensure an accurate representation of the time it takes to do something is to time repeated runs and consider the average.

In addition, to ensure accuracy, the database of the Web applications using different platforms will have the same structure and the same number of records. Moreover, the

Web applications will also be run with a different number of records to test their scalability.

The same computer serves as the client and the server to minimize any possible effect of congestion on the university network. Moreover, the Web applications will be executed after minor changes have been made to their programs and the computer has been restarted, to ensure not only that they have been re-compiled and not just re-loaded from the cache memory, but also that the memory is free.

1.8.2.2 Scope and assumption of the latency experiment

The programming for the Web applications will be limited to data access from a relational database. This is because database access is the most common and most important form of utilization of Web applications (Adida, 1997b).

1.8.3 Overall analysis in the testing of the framework

In testing the framework, we use weights as a means of varying the importance or relevance of each major category of criteria and for each criterion also. A weight is attached to each of the criteria as well as the programming experiment, so that the user has the choice of excluding any of them by using a weight of zero (0) or to increase their relative importance by increasing each weight relative to others.

At the lowest subdivision of the criteria, there is a series of questions that are used to evaluate the platforms. Each question has a relative weight such that users can use a weight of zero (0) or one (1) to exclude or include the question respectively.

The total scores will be calculated for each dynamic Web platform on which the framework is tested. This will consist of the various scores obtained using the different criteria categories, as well as the score assigned in the experiment. Weights assigned by the users will be used to multiply the scores. Implementation of a tool to use the framework on a Microsoft Excel spreadsheet allows users to vary the weights of the

various criteria and the experiment, and to calculate the total scores automatically. The platform with the highest total score is the selected platform.

1.9 Summary and outline of the remainder of the dissertation

Chapter 1 presents the background and the motivation for the study. The widespread use of Internet and the World Wide Web (WWW) has led to the availability of various platforms for creating dynamic contents for Web applications. An institution that teaches Web application development must use appropriate platforms for teaching desired concepts. In order to enhance productivity within the limited practical class, such a platform should enhance the comprehension of identified concepts, easy to learn and use, reduce processing delays, affordable and portable.

However, to determine such a suitable one is not simple. Therefore, this study seeks to provide a solution by the development of a framework to scientifically determine a suitable platform for teaching Web application development in tertiary institutions.

Chapter 1 also presents the blueprint for finding the answer to the research problem. It outlines the approaches used and the steps taken, as well as the enhancement of the reliability of the methods applied.

In progressing from the background to the rest of the study, chapter 2 contains the literature review. It presents various previous studies that analyze and compare programming languages and dynamic Web platforms with or without the use of some set criteria. This is followed by the perspectives and results of related work on lowering the barriers to programming, and we motivate why these aspects should be considered in the framework. Lastly, it highlights why and how this study differs from previous studies.

Chapter 3 identifies and establishes those concepts of Web application development that should be part of the curriculum of a typical second year student at a tertiary institution specializing in Web application development. This will enable us to identify criteria for evaluating the suitability of the platforms for teaching identified concepts. Chapter 3 also

identifies other constraints that the framework should satisfy such as the ease of use, ease of learning, speed of use, performance or latency, cost or affordability and portability.

In chapter 4, we developed a framework consisting of sets of criteria sorted under various categories. The various criteria for evaluating the suitability of the platforms involves identifying measurable qualities that will ease the teaching of the various concepts of Web application development identified in chapter 3. It involves also the qualities that will ensure that the platform satisfies other necessary constraints such as being easy and fast to use by second year students. The criteria to ensure that the platform is fast are complemented by the design of the latency experiment.

Chapter 4 also establishes the criteria for ensuring that the platform is portable and affordable and that it will enhance students' individual learning. Lastly, the chapter presents a summary of the framework and a tool to implement the framework on a spreadsheet.

Chapter 5 presents the results of the testing of the framework with specific reference to the requirements of the curriculum at the Tshwane University of Technology (TUT). This involves the evaluation of four selected platforms using the criteria set to evaluate their suitability for teaching the identified concepts of Web application development at TUT.

Chapter 6 presents the results of testing the framework, using the criteria set to evaluate the ease of use, speed of use, affordability and portability of the dynamic Web platforms, according to TUT requirements. The first section considers the results based on criteria to ensure the availability of desirable qualities of a smart integrated development environment for the platform.

The next section of chapter 6 considers results on qualities that will simplify execution and increase the latency of programs and the results of the latency experiment to estimate the execution speed of programs written using each of the dynamic Web platforms. Lastly, it contains a section that presents the results on the affordability, architectural neutrality and portability of the platforms.

Finally, chapter 7 concludes the study. Here, an overview is given of the study. We summarize the developed framework. This is followed by a combination of the various results leading to the conclusions and recommendations of the study. The research contributions and their implications are discussed and evaluated. The limitations of the study and potential areas for further research are also discussed.

Appendix 1 contains a discussion on the different dynamic Web platforms that can be used for teaching Web application development, and the rationale for the selection of four of the platforms for testing the framework. Appendix 2 contains an informal survey that was used to document the experiences of past students of the four selected platforms, and to determine whether these results corresponded with the results of testing the framework.

The next step is to determine from the literature how comparisons of dynamic Web platforms as well as the comparisons of similar programming languages have been done in previous studies. Did the previous studies establish frameworks before comparing the platforms? Are the comparisons tailored to specific uses such as teaching at tertiary institutions? Have they taken into consideration the concepts that should be taught? Answers to these questions will allow us to identify why this study is unique and necessary, despite previous related studies, as well as to establish necessary criteria as part of the framework being developed. The next chapter discusses these issues.

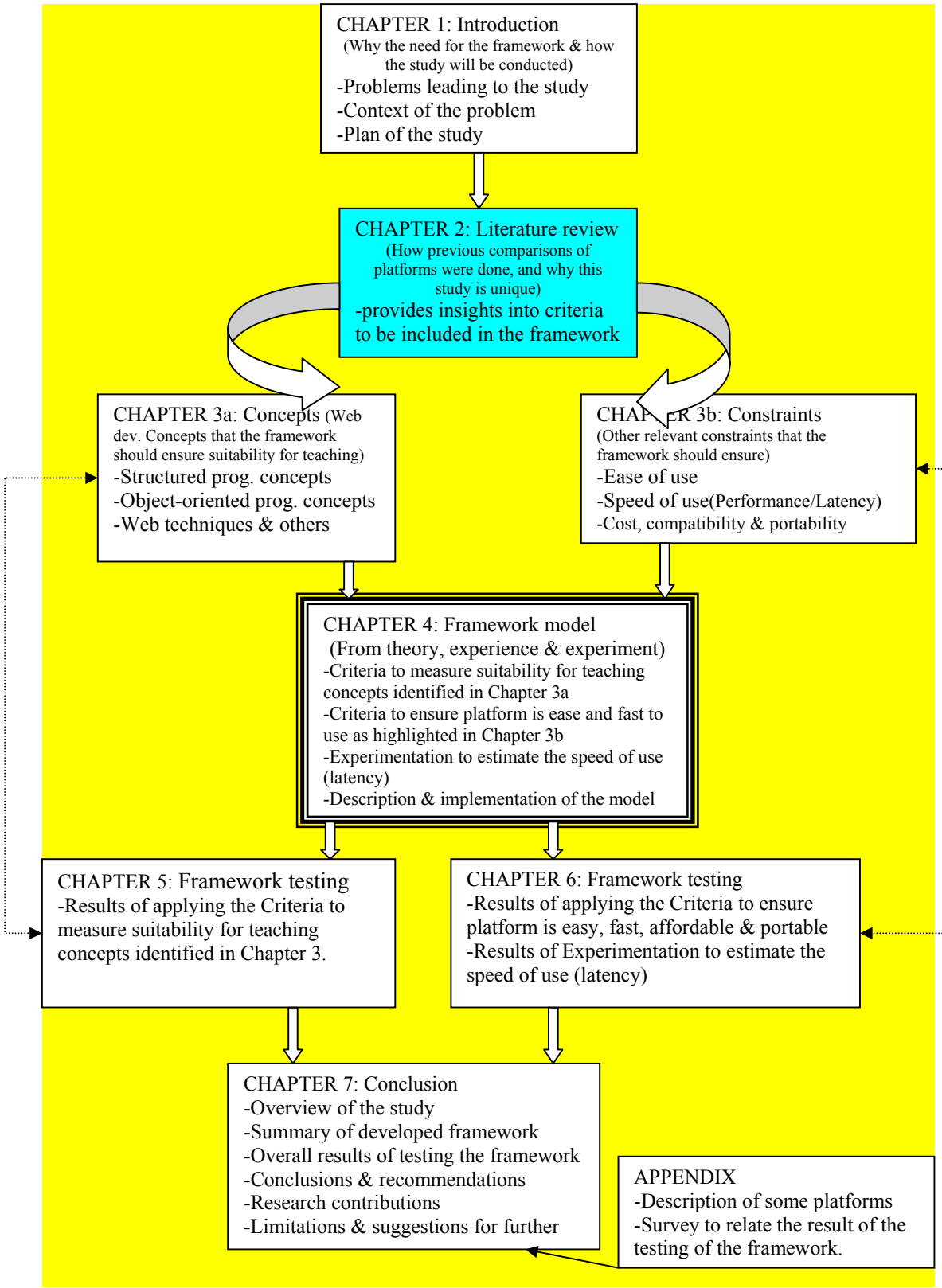

Figure 2-0. Dissertation map highlighting the relativity of chapter 2 to the rest of the dissertation.

CHAPTER 2

2 LITERATURE REVIEW

2.1 Introduction

The previous chapter established the context of the research problem as well as the plan of action to develop a framework for providing answers to this research problem. To further assist in establishing the framework, this chapter reviews various literature and discusses previous studies to provide insights and to establish why this study is unique and necessary, despite previous related studies. Figure 2.1 gives a pictorial outline of this chapter. The overall purpose of this chapter, according to Marshall and Rossman (1989), cited in Creswell (2003), is to relate the study to the larger ongoing dialogue in the literature about a topic, filling in gaps and extending prior studies[3].

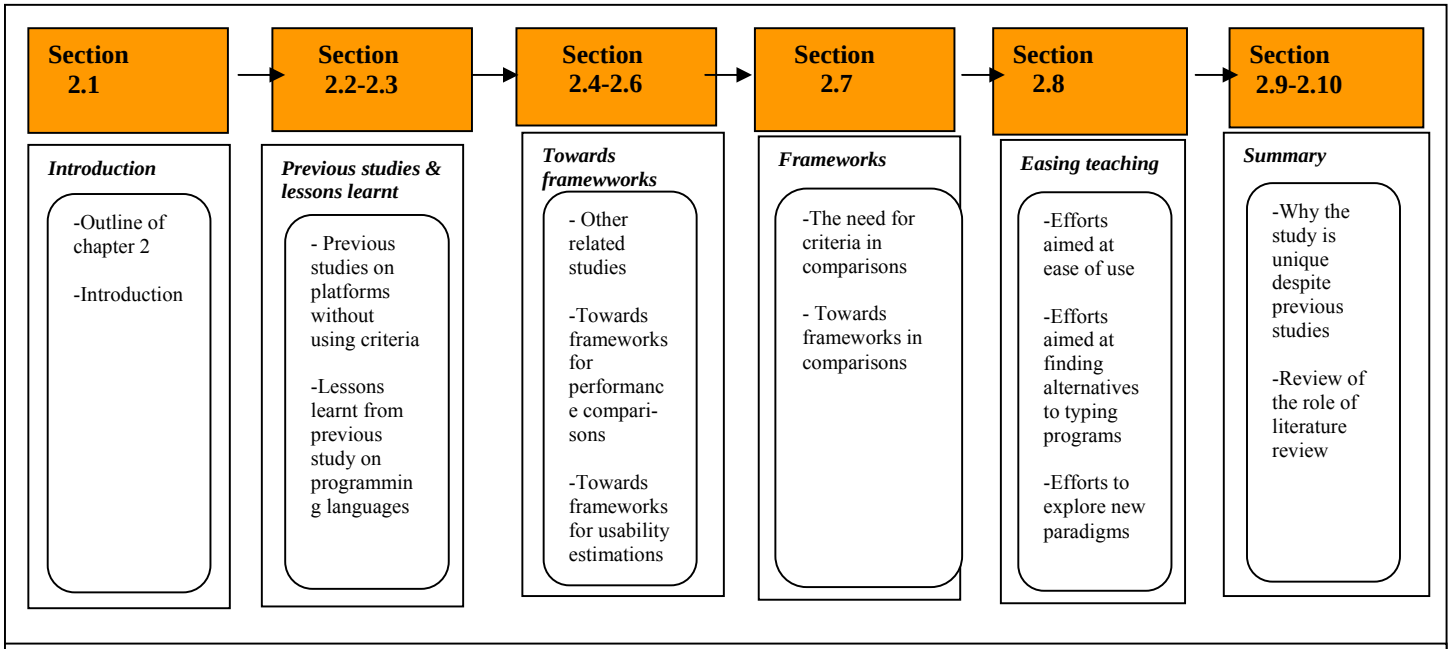

Figure 2-1. Pictorial outline of chapter 2.

[3] In addition to helping one to pin down a research problem, a literature review has numerous other benefits. These are used as a form of review at the end of this chapter. Simply put, the more one knows about perspectives related to one's topic, the more effectively one can tackle one's research problem.

The chapter began with a discussion in section 2.2 on studies that compare platforms without using any factor or criterion. This is followed by section 2.3 that presents the lessons learnt from author's previous related studies. Next, in section 2.4 we discuss studies that emphasize smaller code in their comparisons. In falling short of the standard of using a specific framework, section 2.5 illustrates how performance has been overemphasized in various comparisons as reported in various previous studies. Similarly, section 2.6 examines studies towards frameworks for usability estimations and relates these to the focus of this study.

This leads us to section 2.7 discussing studies that attempt to use some criteria or studies proposing or highlighting the use of a few factors in their comparisons. Other related studies involving platforms and programming language comparisons are also presented. These enable us to identify criteria that we can include in this framework, and led us into studies that recommend a comprehensive framework for determining platforms. At this point, we highlight the need for frameworks that are more specific and relevant for the purpose of choosing the platform.

To further identify more criteria that could be included in our framework, we present in section 2.8 various studies aimed at easing the teaching of programming. Lastly, section 2.9 to section 2.10 summarize, review and emphasize the uniqueness of this study.

2.2 Previous comparisons of platforms without using criteria

Various previous studies have compared dynamic Web platforms and similar programming languages. However, most of these studies did not use any criterion as a basis for the comparisons. Kruse (2003) illustrates the differences in the strengths and weaknesses of Personal Home Page (PHP) and Java 2 Enterprise Edition (J2EE) in tabular form. Below is a table adapted from Kruse (2003). This comparison uses

terminology or factors such as ease of learning, scalability, availability, online support, availability of books and other references, functionality, cost of servers, and availability of tools to configure server and runtime environment. However, these factors are not rigorously investigated and presented for the two platforms compared.

Table 2-1. Strengths and weaknesses of PHP and J2EE as adapted from Kruse (2003)

	Strengths of PHP	**Strengths of J2EE**
	Easy to learn	Scalability
	Free	Lots of functionality
	Dedicated Web development tool	Excellent load balancing
	Platform independent	Platform independent
	Lots of support available online, in form of forums, discussion groups and tutorials. This is due to its popularity amongst developers.	Good books available as well as tutorials
	Executes fast	Stable
	Light on resources when level of Traffic is low	Handles heavy traffic well
	Weaknesses of PHP	**Weaknesses of J2EE**
	Heavy on resources when high level of traffic is encountered. This is because for each client, the script has to be loaded into memory.	Needs expensive Servers to run
	Less functionality than Java	Steep learning curve

Klopper (2003) compares Personal Home Page (PHP), Active Server Page (ASP) and Java Server Page (JSP) in terms of their advantages and architectures. As in the case of Kruse (2003) above, comparison must be based on a variety of factors supported by scientific facts and results. This is in line with Ashenfelter's (1999, p.105) assertion that before analyzing tools, it is worth discussing how to evaluate them.

Bishop and Hurter (1999) examine some of Java's competitors, namely the Scripting languages: Tcl/Tk, Perl and Python, with the following results. Python incorporated the features of Modula-3 into its scripting syntax thus making it suitable for "programming in the large", unlike Tcl and Perl. Some of the distinguishing features of Python as reported by Bishop and Hurter (1999) are:

- It is an interpreted, byte-compiled, interactive, object-oriented programming language.
- It incorporates modules, exceptions, dynamic typing, very high-level dynamic data types, and classes.
- It is a procedural programming language that is also good for GUI programming.
- Programs written in Python are typically much shorter than equivalent C or Java programs for several reasons:
  - ⇒ The high-level dynamic data types allow one to express complex operations in a single statement.
  - ⇒ Statement grouping is done by indentation instead of begin-end brackets.
  - ⇒ No variable or argument declarations are necessary (Bishop & Hurter, 1999).

Bishop and Hurter (1999) also did not use a pre-established set of criteria or framework for the comparisons. However, some of the findings could be useful in formulating the criteria in our framework.

In the survey of middleware platforms by Cooper (2001), it was concluded that ColdFussion is fast to learn and fast to use, and CGI is also fast to learn. He mentioned that Servlets are difficult to learn and use - even by someone who already knows Java. This is based on intuition rather than scientific facts. The results, however, correspond with those of Dehinbo (2004b) which motivated this current study, and in which Java's highest Line of Code for a simple solution to the given problem is perceived as making it more difficult to learn.

Dehinbo (2000), one of the author's previous research study unravels the communication path between the Java programming language, Web server and any modern relational database. Similarly, Dehinbo (2003a), a study to enhance quality of Internet knowledge of Students in TNG by using a Web-based online assessment system, utilized the Java Servlet technology to implement the online tests and assignments which, when completed by students from any computer connected to the internet, communicated the answers directly to the lecturer's computer / server in the form of a text file.

From these studies, the author can confirm Cooper's (2001) statement that "Servlets are hard to learn and use - even by someone who already knew Java". This is, however, simply based on intuition. Scientific evidence is required to support this assertion.

In order to procure more scientific evidence to support comparisons, the author conducted some further studies. The next section presents a study by the author which attempts to provide some scientific evidence such as calculating the Lines of Code (LOC) of programs written in various languages and using surveys to obtain information from experienced users of such languages and platforms.

2.3 Lessons learnt from author's previous study on programming languages

As mentioned in the background in chapter 1, the previous work that led to this study is Dehinbo (2004b), which was inspired by the low pass rate in programming language subjects at the former Technikon Northern Gauteng (TNG). Dehinbo (2004b) aims to identify a suitable programming language for the new B.Tech. Information Technology programme which could be offered in the Computer department of TNG.

The study involved investigating the ease of learning, ease of use under pressure, flexibility and suitability for complex jobs as well as the overall rating for four programming languages, namely, Pascal, C++, VB and Java. A survey technique using questionnaires was used, supplemented by an experiment to estimate complexity through the LOC estimation for a simple programming exercise written in the various languages.

A major finding of Dehinbo (2004b) was that no single language adequately satisfied all the criteria. From the study, C++, Java and VB are all recommendable. VB is recommended for its ease of use under pressure due to the "drag and drop program generating" facility. C++ is recommended because of the fact that for simple programs, C++ has few LOC, which makes it easer to learn. Java is recommended as it is suitable for complex jobs and flexibility.

Looking back now, it is clear that Dehinbo (2004b) lacks a solid framework that ensures repeatability of the evaluation for other relevant programming languages. Most especially, being based largely on the perceptions of respondents in a survey limits the scientific rigour that could be of great value. However, to add to the scientific rigour of this current study unlike as in of Dehinbo (2004b), a survey is included in the appendix for information purposes only, just to document how it relates to the results of testing our framework.

This current study supports the importance of the LOC estimation as a measure of the readability of code. However, in this current study, rather than simply estimating LOC, we look more deeply at incorporating into our framework, criteria that would make code smaller, such as the use of higher level functions and more compact operators. In support of better readability of code resulting from criteria to ensure smaller LOC, the next section presents some other related studies.

2.4 Other related studies emphasizing smaller code in their comparisons

In an empirical comparison of seven programming languages, Prechelt (2000) observed that designing and writing programs in the scripting languages, namely Perl, Python, Rexx or Tcl, takes no more than half as much time as writing it in C, C++, or Java. Moreover, the resulting program is only half as long. He therefore concluded that the scripting languages offer reasonable alternatives to other full programming languages, and they may offer significant advantages with respect to programmer productivity for reasonably small programs.

Bishop and Hurter (1999) also confirmed that a Java version of a server program in Bishop (1998) is nearly four times as long as its Perl's version. This is assumed to make Java difficult to learn.

The lesson learnt here is simply that platforms that enable programs with smaller Lines of Code will be easier to learn, but this needs to be within the context of other relevant criteria using a comprehensive framework based on scientific facts. This opinion will be incorporated in the framework being developed.

Another concern about Prechelt's study is the cross platform comparison of programming languages with dynamic Web platforms. Moreover, the focus of Prechelt's study is not clearly aimed at determining suitability based on specific use, such as teaching specified concepts. This is why we have devoted major parts of chapter 3 to the identification of Web application development concepts that should be taught, and our framework will then include the suitability of the platforms for teaching these concepts.

However, as illustrated in the next two sections, various previous studies utilize one criteria or another in their comparisons. Most studies that compare platforms concentrate on performance criteria (Vinoski, 2003) although some measure performance using various factors. In the next two sections, we will examine the move towards frameworks for performance and usability comparisons in some previous studies, and highlight the need for the use of more comprehensive criteria.

2.5 Towards frameworks for performance comparisons

Vinoski (2003) realizes that various comparisons of programming languages concentrate on performance comparisons. Renaud *et al* (2003) reported on the works of Hsier and Sivakumar (2001) and Shousha *et al* (1998), stating that the measurement of software performance by and for experts is a well known task. Renaud *et al.* (2003) note from other studies that various metrics can be used to measure performance of algorithms in distributed systems, namely: response or waiting time, Synch delay, number of messages exchanged, throughput, communication delay, node fairness, Central Processing Unit (CPU) cycle usage, and memory usage.

Renaud *et al.* (2003) state that since no single metric can be optimal for all applications, it is necessary to ensure that the developer can obtain metrics which reflect the need, priorities and workloads of the particular distributed system. The first four metrics were most suited to measuring algorithm performance specifically. The fifth metric is more dependent on network load than a specific algorithm, the sixth is difficult to quantify and the seventh and eighth produce measurement of debatable merit in judging algorithm efficacy (Renaud *et al.*, 2003).

In this current study, the performance metric included as a criteria in the framework being developed is the response or waiting time, which has also been used by other researchers. For example, in his survey of dynamic middleware systems, Cooper (2001) estimated the response time for Personal Home Page (PHP), Active Server Page (ASP), ColdFusion and Java Server Page (JSP).

Marshak and Levy (2003, p.3) propose a new approach to estimating user perceived latency that is based on server side measurement. This approach uses a new technique in which a tiny and practically unnoticeable zero sized inline HTTP object, called the sentry, is placed at the end of the HTML document so that it does not add overhead. This tracks the arrival time at the user. Marshak and Levy (2003, p.3) evaluated platforms only in terms of user-perceived latency.

It should be pointed out however, that performance has somehow been overemphasized in various studies. This view is shared by researchers such as Vinoski (2003), who agrees just as the author agrees in this current study that a suitable framework for comparison should involve other relevant factors. Vinoski (2003) indicates that while dynamic Web platforms have a variety of qualities such as size, cost, complexity, flexibility and performance, it is the difficulties involved in thoroughly and accurately evaluating the platforms that tempts people to take shortcuts. Such shortcuts include avoiding the issue altogether by simply buying or adopting platforms based on recommendations from suppliers.

Referring to another shortcut in dynamic Web platforms' evaluation, Vinoski (2003) explains that people check only those qualities that are easily measurable, such as performance. He goes on to say that an interesting side effect of this is that it has unintentionally led many dynamic Web platform users to presume that "high performance" is the same as "high quality". Meanwhile, such works could be entirely meaningless, depending on the nature of one's Web application[6].

Vinoski (2003) also shows that, depending on the application, other qualities such as scalability, flexibility and adaptability, ease of use, tool support, and standards conformance could very well take precedence over performance.

Given the need for our framework to cater for the suitability of teaching second year students, we found it necessary to consider other factors such as usability in the framework. This is in addition to other factors, as the next section will examine the moves towards usability comparisons and present the necessity of adapting this usability concept and adding other useful factors to the framework for the evaluation of dynamic Web platforms.

2.6 Towards frameworks for usability estimation

As pointed out earlier, the implementation of the framework being developed in this study is focused on determining the suitability of dynamic Web platforms for teaching Web applications development. This is closely related to the concept of usability, except that we are not considering the usability of a developed software product but are interested rather in evaluating how usable are the platforms used for developing the dynamic Web applications.

Singh (2001) states that, over the years, several researchers have produced sets of principles or guidelines aimed at improving the usability of interactive systems. These guidelines can take a variety of forms and may be obtained from several sources such as

[6] It is a lot like going out to purchase a new family sports utility and coming home with a Porsche 911 Turbo: it doesn't have room to seat the family, nor is it capable of carrying any cargo, but it is the fastest vehicle you can find (Vinoski, 2003).

journal articles, textbooks, companies' in-house-style guides and so on. These guidelines, listed in Singh (2002), include aspects such as effectiveness, efficiency, safety, utility, learnability, flexibility, robustness, memorability, etc. While some of these can be considered fairly broad, general and very vague, it is still necessary to consider most of them when choosing a tool or platform.

Adeyemo (2000) states that as we enter the new millennium, it really seems to be "Usability or else!" for business. Though he focuses on ease of navigation of Websites, some of the criteria factors are relevant for use in a framework for evaluating dynamic Web platforms. He lists some possible usability measurement criteria adapted from Keller (1990, p.287), which include:

- Percentage of task completed per unit time (speed metric)
- Ratio of successes to failures
- Time spent on errors
- Percentage of competitors that do this better than current product
- Number of commands /statements used
- Frequency of help or documentation use
- Time spent using help or documentation
- Percentage of favourable or unfavourable comments
- Number of good and bad features recalled by user
- Number of available commands not invoked
- Number of times the user needs to work around a problem
- Number of times the user expresses frustration or satisfaction.

These metrics provide a definition of usability against which the website usability can be tested. Some of these metrics will also be adapted for use in the framework being developed.

Wesson and Van Greunen (2002) summarized usability data as comprising the three aspects of the definition of usability; efficiency, effectiveness and user satisfaction:

- Effectiveness relates the goals of using the product to the accuracy and completeness with which these goals can be achieved. Common measures of effectiveness include the percentage task completion, frequency of errors, frequency of assists and frequency of access to help or documentation.

- Efficiency relates to the level of effectiveness achieved to the quantity of resources expended. Efficiency or performance is generally assessed by the mean time taken to achieve the task.
- Satisfaction or acceptance describes a user's subjective level of satisfaction when using the product (Wesson & Van Greunen, 2002).

Dix *et al.* (1998, p.162-174) present principles to support usability in three categories:

- Learnability, referring to the ease with which new users can begin effective interaction and then attain a maximal level of performance. Usability principles related to learnability include predictability, synthesizability, familiarity, generalizability and consistency.
- Flexibility, referring to the multiplicity of ways the user and the system exchange information. Usability principles related to flexibility include dialogue initiative, multi-threading, task migratability, substitutivity and customisability.
- Robustness, referring to the level of support provided to the user in determining successful achievement and assessment of goals. Usability principles related to robustness include observability, recoverability and responsiveness.

For dynamic Web platforms, the focus of usability is on the features of the dynamic Web platforms being used in the design and implementation of the system, since they need to be easy to learn. Some of these metrics above will also be adapted for use in the framework being developed. This is necessary because some dynamic Web platforms or programming languages in general have various features which, as indicated by Sebesta (1996, p.2-3), place limits on the kinds of control structures, data structures and abstractions one can use and the form of algorithms that can be constructed.

Thus, in summary, we consider evaluating usability as part of the framework being developed as necessary for the following reasons adapted from Adeyemo (2000):

- Educating other computer professionals
- Increasing satisfaction
- Decreasing anxiety (which server should I use, which database? etc.)
- Increasing trust of technology
- Enhancing global communication

- Increasing consistency across products
- Decreasing learning time - how long does it take for typical designers to learn to use the language / middleware?
- Increasing performance -
  - ⇒ how long does it take to perform tasks?
  - ⇒ how many and what kinds of errors are commonly made during typical use?
- Increasing retention over time
- Increasing productivity -
  - ⇒ how many Source Line of Code (SLC) can be written per man month?
  - ⇒ how easy is it to understand error messages and how long does it take to work around the error?
- Establishing scalability - is it adaptable to large databases?
- Good development environment and validation tools
- Availability of add-ons e.g. email
- Support for software reuse and maintenance
- Security of information
  - ⇒ does the server allow encryption of user information passed across?

We have looked at various studies that recommend the use of one factor or another in the comparisons of programming languages and dynamic Web platforms. We reviewed studies that compare programming languages and or platforms. Some are based on intuition, while some use one or two factors as criteria. Others, however, have used several factors but have not integrated them into a comprehensive framework relevant to specific need. This study aims at filling this gap in the body of knowledge.

However, this study still argues for the need for a comprehensive framework backed with theory, experience and experiment. In arriving at this conviction, we will next present previous studies highlighting the use of various criteria or a comprehensive framework. This will lead us to the conviction that evaluating platforms should be done scientifically, using established comprehensive yet specific frameworks rather than intuition or feelings or even surveys.

2.7 The need for criteria towards frameworks in comparisons

Various comparisons of programming languages and dynamic Web platforms have been made in the past. Some of these have been based on factors such as usability, ease of use, ease of learning, performance *etc*. Some of these studies propose the use of a combination of two or more factors for their comparisons. There is a progression to studies that use

factors that have been integrated into a more general comprehensive framework for the purpose of the comparisons.

Fraternali (1999) proposes that a suitable platform for developing data-intensive Web applications should cater adequately for security, scalability, availability, interoperability with legacy systems and data, ease of evolution and ease of maintenance. These are in addition to factors that will enhance reuse of code, objects, classes and libraries, and enhance usability as well.

Menasce (2002) indicates that developers adopt frameworks based on quality of service in terms of response time, throughput and availability. Similarly, Maximilien (2004) includes reliability, availability, request-to-response time, interoperability, robustness, integrity, scalability, security, stability, programmer experience and economic cost as factors in the quality of service when evaluating platforms.

Although Van Hoff (1997) evaluates only Java as a programming language, he evaluates it in terms of simplicity, familiarity, and object-oriented features with more emphasis on single inheritance. Other factors used include robustness via type safety, memory management, array bound checking and multithreading, as well as security via access modifiers, restriction on pointer arithmetic, and byte-code verification with access control in a process referred to as sandboxing. Van Hoff (1997) explains how platform independence enhances Java's wide acceptance.

Adida (1997a) evaluates CGI and FastCGI in terms of security, portability and functionality. He notes that, unlike CGI, FastCGI is more secure because it runs in a separate process to prevent it from taking down the server when a process fails. On portability, converting a CGI to a FastCGI is as easy as adding a few lines that call the FastCGI functions. On the whole, Adida (1997a) indicates the improvement on the functionality of CGI by FastCGI.

Hartman (2001) examines some tools for developing dynamic websites, namely ASP, PHP and ASP.NET. He mentions three factors that complicate choosing a scripting

environment that will make the website fast, database-driven, reliable, and created on time and under budget. Two of these factors are given below. First, there is the issue of culture among developers. He believes this has a lot to do with the ideological camps to which they belong[7]. He mentions too that he has encountered very few developers who are equally willing to use both camps, or who can talk about "the other" technology without a trace of disdain.

According to Hartman (2001), the second factor that complicates choosing a scripting environment is that a website's future scalability and functional requirements, although hard to predict, are necessarily a part of the equation. The choice between Java Server Pages (JSP), PHP and ASP (or its successor ASP.NET) might restrict which servers and platforms the site can run on or impact the feasibility of developing future features, such as database-linked connectivity with extranet partner sites (Hartman, 2001).

Hartman's study arrived at these conclusions:

- ASP is a commercial technology while PHP is an open-source technology.
- ASP is somewhat easier to learn whereas PHP enhances flexibility.
- ASP is limited to IIS/PWS on Windows while PHP runs on a multitude of servers and platforms.
- ASP.NET. promises to be a faster and more efficient environment than ASP, and possibly PHP. It makes it easy to create SOAP- and XML-based Web services. But it too is limited to the Microsoft platforms.

In terms of the comparison, it is important to note that Hartman (2001) uses terminologies or factors such as perception of developers, performance and efficiency of platforms, ease of learning, and cost and scalability of the platforms. These factors are not, however, exhaustively investigated.

[7] If they love to tinker with source code because it lets them develop solutions that are more efficient than off-the-shelf products, and if their cubicles are embellished with defaced portraits of Bill Gates, they'll prefer to use PHP. But for convenience and efficiency, they'll prefer to use ASP (Hartman, 2001).

In a study comparing the performance of middleware architectures for generating dynamic Web contents, Cecchet *et al.* (2003) evaluate three dynamic Web platforms, namely PHP, Java Servlets and Enterprise Java Beans (EJB). The study measures the performance of these three platforms using two applications benchmarks: an online bookstore that stresses the server back-end and an auction site that stresses the server front-end. It was found that EJB had higher latency values (worse performance) than both PHP and Java Servlets, while Java Servlets had higher latency values than PHP which has lowest latency giving the best performance out the three (Cecchet *et al.*, 2003).

Cecchet *et al.* (2003) attribute PHP's better performance to the fact that it executes as a module in the Web server, sharing the same process (address space), thereby minimizing communication overhead between the Web server and the scripts. This is unlike Java Servlets, which runs in a Java Virtual Machine (JVM) as a separate process from the Web server and can thus be placed on a separate machine. However, it was observed that the flexible ability of Java Servlets to execute on a separate machine from the Web server and its ability to perform synchronization led to better performance when the front-end was the bottleneck (Cecchet *et al.*, 2003).

In terms of ease of development, Cecchet *et al.* (2003) explain that PHP scripts are easy to write because they can be seen as an extension of the HTML language that embeds code directly into an HTML page. However, they voice the concern that the database interfaces of PHP are *ad hoc* and code maintenance for database is awkward because new code must be written for each new database to which the scripts need access. On the other hand, Java Servlets accesses the database using JDBC which makes it easily portable between databases.

Thus, Cecchet *et al.* (2003) evaluated platforms in terms of performance and ease of development. Their focus however is not aimed specifically at determining suitability based on specific use, such as teaching specified concepts. Moreover, they compared EJB along with other non-EJB architectures. Therefore, Cecchet *et al.* (2003) did not compare platforms with similar architectures, as EJB's architecture is more specialized than that of PHP and Java Servlets.

As part of their Middleware Technology Evaluation (MTE) project, Gorton and Liu (2003) conducted several experiments to explore the performance implications of two common application architectures supported by J2EE's Enterprise JavaBean (EJB) technology. These are the stateless session-bean-only architecture which contains hand-coded JDBC statements for accessing external data sources, and the stateless session façade architecture which uses a session bean as a wrapper for a set of container-managed persistence (CMP) entity beans encapsulating persistent data.

The two architectures were analyzed in terms of ease of development, ease of maintenance, performance and scalability. The session-façade architecture was found to be easier to develop because the EJB container generates the entity beans' data code rather than requiring developers to manually code the JDBC calls. Also, applications based on the session-façade architecture were found to be generally easier to maintain: business logic changes are simple due to the reliance on entity bean interfaces, rather than direct modifications of the more complex session-bean code and JDBC statements (Gorton & Liu, 2003).

However, Gorton and Liu (2003) reported further that the session-bean-only seemed likely to perform and scale better presumably because it uses well-written and efficient JDBC statements with a single bean as a transaction participant. On the other hand, the session-façade alternative incurs more inter-EJB calls with multiple transaction participants slowing the commit protocol. It also uses automatically generated JDBC statements that might not be optimal in terms of performance.

Gorton and Liu (2003) acknowledge other architectural options, including bean-managed persistence and "stateful" session beans, as well as non-EJB alternatives such as Java Server Pages or Servlets which access the database directly. However, their study did not compare these alternatives as this they reported as beyond their scope.

In the process of choosing a language or platform that is usable and suitable for teaching introductory programming, Holt *et al.* (1997) lists the following criteria:

1. It should be appropriate for introducing programming concepts used in the real world such as in business, science and government.
2. It should encourage systematic problem solving.
3. It should be small, convenient and easy to master.
4. It should be easy to implement on economical processors / machines and compilers / interpreters.

Similarly, Kolling and Rosenberg (1996) suggest the following sets of principles when making decisions about language issues:

1. No conceptual redundancy. This is because achieving the same thing in a variety of ways can mean flexibility to the expert, but is usually confusion to the beginner.
2. Clean concepts. The concepts of the language should be presented in a way that directly reflects the theoretical model.
3. Readability. Achievable by favouring expressive keywords. This enables students to learn example programs and to re-read their own programs.
4. Software Engineering support. This is necessary to avail mechanisms and guidelines that support good program development.

Moreover, Hadjerrouit (1998) evaluates the suitability of Java as a first programming language using the following criteria:

1. Programming concepts to be taught. These include problem solving skills, algorithmic thinking and structured programming.
2. Novice usability. How sufficiently simple the language is.
3. Marketability and support for larger programs. This motivates the student because he would want to be taught using languages that sells well in the market.
4. Use in subsequent courses. Knowledge gained should be useful in later studies.
5. Number of programming languages. Reduction in the number of languages to be taught could give the student a broader knowledge of a particular language.
6. Programming paradigm support. The language should support the desired paradigm that students need to be exposed to.
7. Motivation. There should be some enthusiasm about the language.

Although these criteria refer to programming language taught in the first year, the criteria would be relevant for selecting platforms used for teaching Web application development in the second year. Apart from the structured and object-oriented programming concepts that students would have learnt, the remaining concepts would be relatively new and different.

Without doubt, these criteria are vital when considering which dynamic Web platform to use. However, in most cases, these factors have not been integrated into a comprehensive framework that is relevant to the specific purpose of the comparisons, supported with theory, experience and experiment, and tested to demonstrate its applicability. Ashenfelter's study, presented next, attempts to provide a more relevant framework.

Ashenfelter (1999, p.106) provides a set of relevant criteria for comparison by indicating that a Web database tool must be analyzed in terms of its purpose (what it is designed to do), technology (ease of use, robustness, scalability, security, performance, etc.), support (portability, cost, ISP support), and how well it works in the real world. Ashenfelter (1999, p.105) does agree that the importance of the various factors will vary from project to project and often from person to person, but it is nevertheless important at least to consider each of the criteria before making choices of different Web database tools.

Although Ashenfelter's study is presented in a book titled "*Choosing a database for your website*", we consider most of the criteria relevant for use in a framework to determine platforms for teaching Web application development. One problem with Ashenfelter's study is an issue of improper nomenclature. Even the title, "Choosing a database for your website", is rather misleading: in effect, Ashenfelter's study covers choosing Web platforms that will add dynamic database processing to websites.

Also, it is not grammatically correct to group ease of use, robustness, scalability, compatibility, security, performance, reusability and extensibility under "technology". Similarly, portability, listed under the "support" grouping, has a lot to do with compatibility which appears under "technology".

In terms of defining "purpose" as one of the criteria, Ashenfelter (1999, p.106) considers this in terms of what the tool is designed to do, as can be inferred from questions such as; "Is the tool designed for dynamic publishing of Web pages from contents stored in database? Is it for information transactions such as electronic commerce that access the enterprise database(s)? Is it for data storage and retrieval of analytic or historical information? And lastly, is it for Web applications for combining any other database functions with a browser front end?

We should be more specific and consider rather how well the platform serves our own purpose. In other words, apart from being designed for developing Web applications, is it suitable for teaching the development of Web applications? To be more specific, is it suitable for teaching the desired concepts? This is essentially why we devote a major part of chapter 3 to the identification of desired concepts to be taught to second year students of Web application development.

As part of being more specific, some of the factors identified in Ashenfelter (1999, p.106) are no longer relevant, or are at the least less important to our purpose. Robustness and security are more relevant to enterprise applications than to simple applications being developed by students. Moreover, we do not consider factors such as Internet Service Provider (ISP) support relevant since the Web applications being developed by students are done on a local server, and students do not use the hosting services of an ISP.

Moreover, the set of criteria as given in Ashenfelter's study does not constitute a detailed or comprehensive framework. This is because for each of the criteria, there is only one level of evaluation. For example, we believe that purpose as a criterion should cater for suitability to various subsections of the concepts that we want to teach, such as structured programming principles, object-oriented principles, Web techniques, *etc*.

The evaluation of each criterion in Ashenfelter's study is qualitative. This will lead to subjectivity as it will be difficult to compare qualitative data accurately. It will also be difficult to choose a platform based on qualitative data. This will affect the repeatability

of the study. This is why we propose the use of questions for each criterion, such that we can assign quantitative values which add up to one value for each of the platforms. The platform with the highest value will then be regarded as the chosen platform.

Lastly, although Ashenfelter (1999, p.111) agrees that the relative importance of each attribute or criterion varies from project to project and developer to developer, there is no facility for users to vary this degree of importance. This is why we find it necessary to recommend the possibility of attaching weights to the criteria as well as to each question within the criterion.

In all fairness, however, Ashenfelter's (1999, p.105) study is very useful. Some of the criteria identified will be adapted, with our own appropriate nomenclature and modifications, in the present study.

To assist in moving towards the establishment of the framework, the next section presents some studies that identify barriers to programming and pitfalls for beginners, with the aim of using these to ease the teaching of programming. Lessons learnt here will be of great value in shaping the contents of the criteria in our framework.

2.8 Previous works aimed at easing teaching of programming

The previous discussion highlights frameworks and factors involved in the evaluation of Web platforms and programming languages with the goal of using them in determining a suitable platform for teaching second year students. To further assist in this goal, this section discusses previous works that highlight problems in teaching some programming languages and how these problems have been used either in helping to choose or to design more suitable programming languages and platforms.

These problems must be recognized in order to aid learning and teaching in an effective way (Lahtinen, Ala-Mutka & Jarvinen, 2005). As supported by Biddle and Tempero (1998), we believe this will be of use to anyone considering choosing programming language and platforms. However, since Holt *et al*. (1977) highlighted the need to avoid

the temptation to invent YATL (Yet another Teaching Language), which is beyond the scope of this study, we simply need to use the lessons learnt from this review as a guide in the development of criteria in our framework.

Kelleher and Pausch (2005) highlight the challenges of teaching programming languages to beginning programmers who have to learn rigid syntax and rigid commands in addition to learning to solve problems and understand program execution. To subdue these challenges, Kelleher and Pausch (2005) further indicate that researchers have designed and improved various teaching platforms with beginners in mind. These systems are aimed at making it easier to get started on programming and to give students a background that makes it easy to move to more general-purpose or specialized platforms.

The following are examples of such previous works. Though some of these references are relatively old, we believe that we can still learn from them. Some of these works will enable us to include relevant criteria in our framework.

The BASIC programming language was developed as a result of the problems encountered teaching FORTRAN and ALGOL which were considered large and too complex for non-science students (Kurtz, 1981, p.76). BASIC was therefore designed to support a small set of instructions without unnecessary syntax[8]. Similarly, Holt (1977) reveals the removal of redundant structures, inconsistencies that go against students' intuitions, as well as constructs that are easily misused, such as pointers, from PL/1 to form SP/k which is taught incrementally to enhance assimilation.

As in the cases above, the Turing language was designed to contain all the features of Pascal, adding other relevant features but simplifying the syntax by removing the requirements for headers declaring the name of the program and semicolons at the end of each statement (Holt & Cordy, 1988). In essence, Turing was designed to have BASIC's clean interactive syntax, Pascal's elegance and C's flexibility (Holt & Cordy, 1988). Needless to say, it is now a well known fact that Java contains all the features of C++ as

[8] Statements such as LET S = S + 1 are considered easier to understand than S = S + 1 which can be argued as impossible by someone from a mathematical background (Kurtz, 1981).

well as other relevant features, but simplifies the syntax by removing difficult concepts such as pointers (Bergin, 1996; Hadjerrouit, 1998; Van Hoff, 1997).

Unfortunately, even Java is reported as having redundancies with type names, class names and instance variable names cluttering declarations without adding any extra information (Black, 2004). This could account for the design of another language, JJ (Junior Java), to remove much of the syntactic complexity, allowing students to focus more on problem solving and the concept of programming rather than on syntax. In addition, JJ removes much of the punctuation such as braces and semicolons, reduces the number of data types to one, i.e. integer, and has only one way to create a comment (Kelleher & Pausch, 2005).

The author takes cognizance of the "small is beautiful" concept (Black, 2004), or the "less is better" concept (Holt *et al.*, 1977), so that students have less to memorize and the compilation is more efficient. This is similar to the case of the Blue language, as in JJ, which was designed with the criteria that there should be only one way to do everything, such as single loop structure, thereby making its programs readable and easy to understand (Kolling & Rosenberg, 1996). Black (2004) sums these up by stating that one of the great strengths of Smalltalk is that it is small.

In a quest to avoid unfamiliarity of commands, Kelleher and Pausch (2005) report the development of the Grail language which is governed by the principle of single "English-like" syntaxes. For example, the language uses "x" for multiplication to make it familiar to novice programmers with a mathematical background[9].

The lessons we can learn so far from this is that our proposed criteria should emphasize the need for a reasonable number of commands, and these commands should be more explanatory. Commands should have meanings close to their English meanings and operators should have meanings similar to their use in arithmetic and mathematics. We

[9] The Grail language is also reported as using arrows to assign values e.g. a ← a + 1 since a = a + 1 is incorrect and impossible from a mathematical point of view (Kelleher & Pausch, 2005).

do, however believe that in addition, we need criteria to avoid the inability to solve non-trivial problems as in the case of BASIC highlighted in (Holt *et al*., 1977).

In an attempt to prevent syntax errors, some work has been done on producing programming environments for existing languages. The Cornell Program Synthesizer was designed to edit, execute and debug programs in an environment that acknowledges and reinforces the hierarchical nature of programs[10] (Teitelbaum & Reps, 1981). The Gnome environment was designed to display programs hierarchically and as programmers navigate through the programs using arrow keys, it prevents moving on until the program is syntactically correct (Miller *et al.*, 1994). This was later improved upon by the design of MacGnome which allows programmers to navigate by pointing and clicking with a mouse (Miller *et al.*, 1994).

Other efforts were aimed at finding alternatives to typing programs by creating objects that represent code and can be modified using the actions of the programmer within the interface (Kelleher & Pausch, 2005). An example is Pict, which allows one to create programs by selecting relevant icons representing commands from a menu screen (Glinert & Tanimoto, 1984). Another example is Drape, which allows one to create programs by drawing pictures from a collection of pictorial icons representing commands on the left side of the screen (Overmars, 2000). Leogo was designed to allow three methods for creating programs including typed syntax, direct manipulation and an iconic language (Cockburn & Bryant, 1997).

Lessons learnt here include the need for criteria that will ensure the availability of a smart Integrated Development Environment (IDE) for the chosen platform. Such IDEs should have the facility to design Web pages on the screen and then allow switching to the code mode.

Some design efforts explore new paradigms for designing, developing and organizing code (Kelleher & Pausch, 2005). Such works include the design of Pascal to support the

[10] The nesting of the THEN and ELSE statements under the IF command will enhance readability (Teitelbaum & Reps, 1981).

Structured Programming concept (Wirth, 1993) and the design of Smalltalk to introduce the object-oriented programming concept (Kay, 1993). Similarly, the BlueJ environment was developed to encourage one to create and test individual classes rather than complete programs as in Java IDEs (Kolling *et al.*, 2003).

Lessons learnt here include the need for the criteria to ensure that the chosen platform will support both structured programming as well as object-oriented programming concepts. Criteria should also ensure independent testing of modules and classes so that students can see their actions before incrementally combining several modules and classes to form a larger solution.

Yet other efforts have been made to understand program execution as Kelleher and Pausch (2005), Lahtinen, Ala-Mutka and Jarvinen (2005) as well as Milne and Rowe (2002) all indicate that beginning programmers find it difficult to understand how programs are executed and this makes it difficult to find logic errors in their programs. Kelleher and Pausch (2005) report the design of Atari 2000 BASIC which allows one to monitor the program execution by dividing the screen into regions. These contain the program, the variables and their values as they change (Eisenstadt, 1983), the diagnostic errors as well as the output. Lessons learnt here include the need for tracing facilities in the IDE required by the framework.

Lastly, other works have addressed sociological barriers to programming. Kelleher and Pausch (2005) indicate that barriers such as the lack of getting help and ideas from other programmers can make programming very frustrating to students. This informs the need for the criteria to ensure adequate support group as well as free standard documentation.

2.9 Summary and conclusions

In any trade, productivity depends partly on the quality and relevance of the tools being used. The relevance of a tool depends on the features that make such a tool suitable for particular problem situations. Choosing the appropriate tool involves detailed evaluation

of the various options. This implies the need for exhaustive evaluations based on various relevant criteria that are backed by scientific facts and results.

This chapter highlights the use of various criteria in some of the previous studies that have evaluated and compared programming languages as well as dynamic Web platforms. The various studies by the author and their results as well as lessons learnt from their post analysis have also been presented. Some comparisons are based on intuition, while others use one, two or more criteria. In some cases, the recommended criteria have not been exhaustively tested. Others, however, have used criteria that were not integrated into a comprehensive framework relevant to a specific need. This study aims at filling this gap in the body of knowledge.

Although the literature review is very useful in understanding the contexts of the proposed study, giving directions to the study and fitting it into the gaps in the current body of knowledge, it is important to point out that, while the various related studies discussed above are relevant, and have thrown valuable light on this study, they differ from it in the following ways:

This study does not agree that it is sufficient to list the advantages and strengths of each dynamic Web platform. Rather, the advantages and strengths of each middleware system should be examined and ranked in the light of certain desired qualities relevant to that specific “school of thought”. This is because certain strengths may be of utmost priority to one particular “school of thought” and not to another. For example, in Technikons training middle-level workers, the ease of learning of a programming language increases our productivity more than the flexibility of the programming language or platform.

This study considers it very important to integrate into one framework the various factors or criteria that are useful for the evaluation and comparison of platforms. Also, in the testing of our framework, we consider it necessary to exclude EJB because its architecture does not place it in the same category as PHP and Java Servlets, as it is more of an enterprise multi-server platform. Instead, we will include JSP and ASP, which both embed code directly into an HTML page like PHP. Moreover, ASP has the backing of the

VB Script language just as JSP and Java Servlets have the powers of the Java language, and PHP also has full programming language capabilities.

However, other studies such as Holt *et al.* (1997), Kolling and Rosenberg (1996) and Hadjerrouit (1998) have established some criteria for analyzing programming languages and platforms with the goal of determining those most suitable for teaching undergraduates. None of these criteria are integrated into a comprehensive framework peculiar for determining suitable platforms for teaching Web application development.

Unlike most other studies presented, Ashenfelter's study has been found to be most useful and educative for this study. However, problems identified in Ashenfelter's study include misleading nomenclature and the generality of some criteria. The author finds it necessary, therefore, to adapt some of the criteria identified in Ashenfelter (1999, p.105) with our own appropriate nomenclature and modifications.

This study addresses the need for the framework to be specific and yet comprehensive. It also addresses the variability of the degree of importance of the different criteria. The development and implementation of a tool to apply this framework will allow users to allocate weights to the criteria and the questions within each criterion and also to vary these weights and observe the effects on overall results.

Most importantly, unlike other studies reported, this study attempts to solve the research problems using literature study and experiment, thereby exploiting the strengths of both quantitative and qualitative methodologies. In conclusion, by being unique, this study provides an addition to the body of knowledge in terms of the framework developed for choosing appropriate dynamic Web platform to enhance the teaching of Web application development in tertiary institutions in South Africa. This provides the possibility of increasing the learners' potential in such a way that would lead to higher productivity in the software development and services industries in the future.

Finally, to assist in moving towards establishing the framework, studies such as Kelleher and Pausch (2005) as well as Biddle and Tempero (1998) have identified some barriers to

programming and pitfalls for beginning programmer . Moreover, Kelleher and Pausch (2005) led us to a search for other studies that have designed various systems to solve the barriers to programming. These have enabled us to identify and confirm certain criteria that will be relevant for incorporation into our framework.

2.10 Review and link to other chapters

This chapter presents a detailed literature review background. We also present frameworks with various metrics that have been used to analyze dynamic Web platforms. The chapter finally describes various related works, pointing out the unique nature of this study and its contribution to the body of knowledge. Without a doubt, the literature review fulfilled some of the benefits as enumerated by Leedy and Ormrod (2001, p.70):

- ✓ It has increased the author's confidence and professional competence on this topic and area of research.
- ✓ It has served as an exploratory process, which has provided new ideas and approaches. Such new ideas and approaches include some criteria that will be included in the framework.
- ✓ It has informed one about other researches being conducted in this area. This will be useful for future collaboration efforts, and for getting advice and feedback.
- ✓ It has shown how others have handled methodological and design issues in studies similar to this.
- ✓ It has revealed sources of data of which one was not aware of.
- ✓ It has introduced one to measurement tools that other researchers have developed and used effectively.
- ✓ It has revealed methods of dealing with problem situations that were similar to the difficulties being faced by this study.
- ✓ It will help to interpret, compare and make sense of our findings and, ultimately, to tie our results to the work of those who have preceded us.

These benefits of the literature review have shown that the need for such detailed planning before the collection of data cannot be overemphasized. With such planning, we

are in a better position to proceed with the establishment of criteria for the framework. One of the most important criteria is that the framework should ensure that the platform is suitable for its intended purpose. Therefore, the next chapter will identify desired Web application development concepts that can be taught to second year students, so that we can develop criteria that will ensure suitability for teaching these concepts.

It is also important that the framework enables the determination of a platform that will be easy to learn, easy and fast to use by second year students. The last section of the next chapter takes a look at other relevant constraints that the framework should comply with. This will also enable us to establish criteria and experiment to satisfy the constraints.

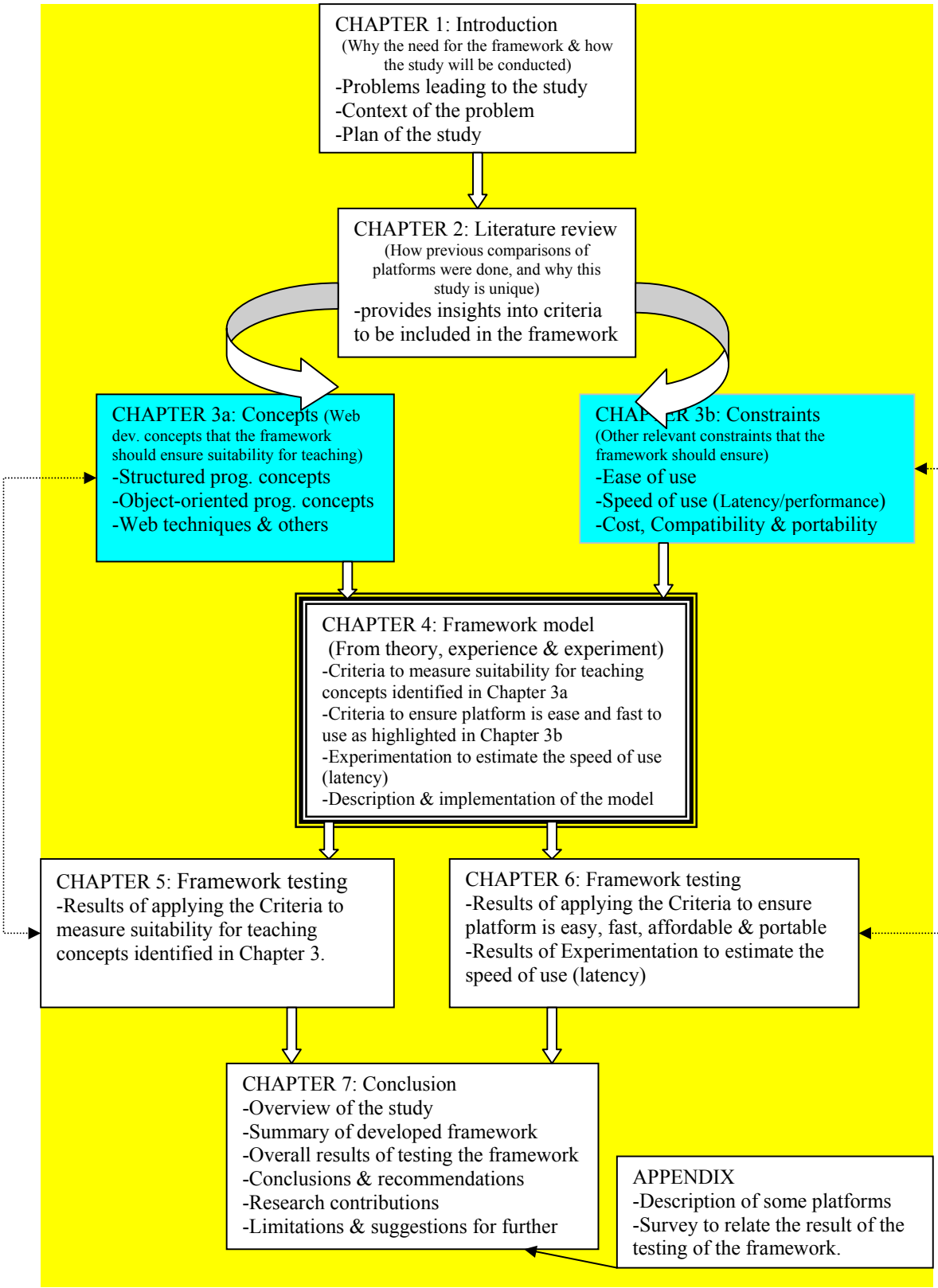

Figure 3-0. Dissertation map highlighting the relativity of chapter 3 to the rest of the dissertation.

CHAPTER 3

3 THE DESIRED WEB APPLICATIONS CONCEPTS TO BE TAUGHT AND OTHER CONSTRAINTS TO BE SATISFIED

3.1 Introduction

In order to enable us establish the criteria for our framework in chapter 4, the purpose of this chapter is to identify relevant concepts of Web application development so that the criteria to be established can ensure suitability for teaching the identified concepts. Furthermore, this chapter also discusses other constraints to be satisfied by any platform selected using our framework. Figure 3.1 illustrates the pictorial outline of this chapter.

Web application development is a specialization of general software development. It will thus involve the utilization of established concepts of general-purpose programming, object-oriented programming as well as some specialized concepts peculiar to the Internet, Web Servers and the Hyper Text Transmission Protocol (HTTP).

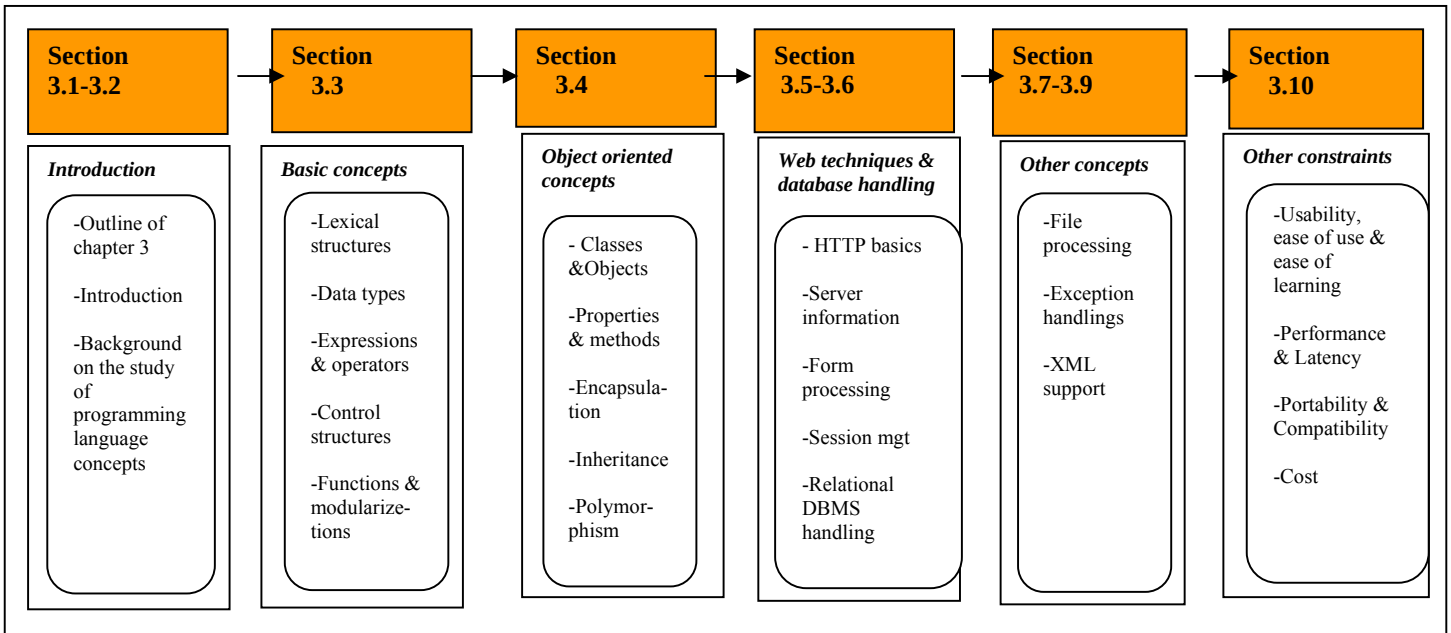

Figure 3-1. Pictorial outline of chapter 3.

The relevant concepts identified are those that should be taught as part of the curriculum for second year students in tertiary institutions specializing in Web application development. This will therefore involve a reinforcement of the concepts taught and clearly understood by a typical first year student, and an introduction to current features that are applied on various applications on the World Wide Web.

As illustrated in figure 3.1 above, this chapter is preceded by a brief background on the need for a sound knowledge of programming concepts in section 3.2. The chapter further contains six sections corresponding to the various concepts to be taught.

Each section begins with the importance of teaching this concept, and then proceeds to how the various components of the concept will be taught. In addition, some questions which students should be able to answer successfully in order to demonstrate an adequate understanding of the contents of the section are presented.

Section 3.3 presents a brief reinforcement of the basic language and structured programming concepts. This is followed by section 3.4 on the object-oriented concepts. Section 3.5 contains the basics of the HTTP, as well as peculiar techniques for static and dynamic Web page processing.

Given the importance of databases and files section 3.6 present the remote handling, storage and retrieval of information on databases while section 3.7 discusses processing on text files. Section 3.8 discusses Exception handling containing details on how to intercept run-time errors and unexpected situations, take corrective measures, and record errors in log files. Section 3.9 covers details of how to generate and extract XML documents in order to present a common format for data transfer.

Section 3.10 discusses the constraints to be satisfied by selected platform using our framework, such as ease of use and ease of learning, performance / latency, affordability, and portability. The chapter is concluded with a summary and a link to the next chapter.

3.2 Background

Generally, programming languages are used as a vehicle for conveying the concepts of programming and the principles of software design. The necessity of understanding programming language concepts is highlighted by the following statement:

> The study of programming language concepts builds an appreciation for valuable language features and encourages programmers to use these features. (Sebesta, 1996, p.3)

Various programming and scripting languages are presented in a typical three-year diploma in Information Technology being offered at Technikons and Universities of Technology. In the first year of study, programming languages such as C++ and Java are used to illustrate the concepts of structured and object-oriented design paradigms. In the Web application development being taught at the second year, the following are the concepts and principles that we are interested in conveying to the learners.

3.3 Basic language and structured programming concepts

This section contains a discussion on the reinforcement of the basic language and structured programming concepts learnt in the first year. It covers elementary topics such as data types, variables, operators, flow control, as well as modularization using functions coupled with data manipulations such as data conversion, string and array manipulations. Inherent in this is the study of the lexical structure which is the set of basic rules that governs how one writes programs in any particular language.

Given the fact that a thorough understanding of these basic concepts is important for successful development of applications, students will be given a brief revision of the basic language and structured programming concepts learnt in the first year. This will involve teaching students the following:

3.3.1 Lexical structure

This is the set of basic rules that governs how programs are written in any specific programming language. From the author's experience on various programming

languages, and using the structures defined in various texts such as Lerdorf and Tatroe (2002, pp.1-20) the followings are considered very important and will be addressed:

- How to form variable names: What are the reserved keywords that cannot be used as variable names? How long can a variable name be? What characters must begin a variable name and what other characters can be included?
- Case sensitivity: Are the reserved keywords or commands case sensitive? Are the variable names and user defined functions and classes case sensitive?
- Statement punctuations: What is used to end statements? Are multiple statements allowed on a line? What is used to separate statements within such multiple statements? What symbol is used to mark compound statements? Do white-spaces matter or can you spread a statement across any number of lines?
- Comments: What are the comment-styles used?

3.3.2 Data types

Understanding the various *data types* is very important in developing applications. Therefore, students will be exposed to various *data types* supported:

- Primitive data types: Are primitive *data types* such as Integers, Floating point numbers, Strings, Booleans and Arrays supported? Are there various functions for manipulating Strings and Arrays?
- Complex data types: Are other complex *data types* such as classes and references supported?
- What other peculiar *data types* are supported?
- Is variant data type supported so that a variable can be assigned values of different types? Can you mix *data types* in an expression?

3.3.3 Expressions and operators

Knowledge of the following will be relevant:

- How are simple and complex expressions formed?
- Operators allowed: What are the symbols used for arithmetic, logical and relational operators?

- Operator precedence: What is the order in which operators in an expression are evaluated?
- Casting: How is the conversion of a value from one type to another done?

3.3.4 Control structures

Various constructs for controlling the flow of execution of a program will be taught. These will include control structures for:

- Decision;
- Looping; and
- Multiple branching.

3.3.5 User-defined functions and modularizations

Modern programs are usually complex and developed with a team effort. Modularization makes it easy for different members of a team to work on different parts of the application. These parts can be developed as modules or functions (Zak, 2001, p.420).

Functions save development time and enhance reuse of code. Instead of rewriting the code in the functions in different parts of the program, they are simply called to do the required task and return result values. Compile time is also saved because, no matter how many times the functions are called, they are compiled only once (Lerdorf & Tatroe, 2002, p.62).

Functions also improve maintainability, extensibility and reliability. This is achieved by allowing one to fix errors or bugs in one place, rather than everywhere a task is performed. Readability is thereby improved by isolating code that performs specific tasks (Lerdorf & Tatroe, 2002, p.61), making it easier to maintain and extend the system, even in a reliable way.

Special attention will be given to the following:

- How to define functions?
- How to call functions, with or without parameters?

- How to return values to the calling program?
- Determining which folders or directories contain the functions components.

The above will be taught over two weeks, in ten thirty-minute periods per week, at the end of which students should be able to show an understanding of the contents demonstrated by successfully answering questions such as the one given below.

3.3.6 Question

Question 3.3.6.1: The Tshwane University of Technology is offering programming seminars to companies. The price per person depends on the number of people the company registers, as shown in the following table:

Number of registrations	*Charge*
1 – 4	*R1000 per person*
5 – 10	*R800 per person*
11-15	*R600 per person*
16 and above	*R500 per person*

Write a Web-based program that companies can connect to in order to register their staff for the seminar. The program will prompt the user to enter the company's identification name and the number of people to be registered. It will then use a function to calculate the amount owed, which is returned to the calling function, which then displays the invoice containing the company's identification name and the number of people to be registered as well as the amount owed.

3.4 Object-oriented programming concepts

According to Lerdorf and Tatroe (2002, p.140), object-oriented programming opens the door to cleaner designs, easier maintenance, and greater code reuse. Real life objects are used to model problem domain objects, as illustrated by the following statements:

> Unlike the procedure-oriented method of programming, the object-oriented method allows the programmer to use familiar objects to solve problems. The ability to use objects that model things found in the real world makes problem solving much easier. (Zak, 2001, p.420)

This object-oriented method covers basic topics such as Classes, Objects, Properties and methods, Encapsulation, Information hiding, Inheritance and Polymorphism.

3.4.1 Classes

According to Lerdorf and Tatroe (2002, p.141), a class is a template for building objects. Zak (2001, p.421) defines a class as a pattern or blueprint used to create an object. When one designs the program, one has to decide the fields for each data item, and then come up with the functions that operate on those data items. In object-oriented terms, one is designing the class by so doing (Zak, 2001, p.421). The following will be illustrated to students:

- How to translate a structured program into an object-oriented program?
- How to declare a class?

3.4.2 Objects

According to Zak (2001, p.421), the objects used in an object-oriented program can take many different forms: the menus, option buttons, or a representation of something encountered in real life such as a car, a credit card, an employee, or a savings account. Various sources such as Lerdorf and Tatroe (2002, p.141) as well as Deitel and Deitel (2003, p.406) define an object as an instance of a class. Lerdorf and Tatroe (2002, p.141) further explain that an object is an actual user data with attached code. We'll illustrate to students:

- How to instantiate objects of a class?
- How to check for the class to which an object belongs?

3.4.3 Properties and methods

These are sometimes also referred to as attributes and behaviours, data and functions. According to Zak (2001, p.421), every object has attributes and behaviours. The attributes are the characteristics that describe the object, such as the name, the value, the withdrawal limit, or the interest on a savings account. An object's behaviour, on the other hand, is determined by the operations (actions) that the object is capable of performing. For example, deposits, withdrawals and interest calculations are behaviours (methods or functions) applicable to a savings account object (Zak, 2001, p.421). We'll illustrate:

- How to declare a class encapsulating data and methods?
- How to include class constructor functions?

3.4.4 Encapsulation and information hiding through abstraction

According to Zak (2001, p.421), a class contains (or in OOP terms, it encapsulates) all the attributes and behaviours that describe the objects that the class creates, whereas abstraction refers to the hiding of the internal details of an object from the user, so as to prevent inadvertent changes to the object. According to Bishop (1998, p.267), abstraction enables a class or method to concentrate on the essentials of what it is doing (its behaviour and interface) and to rely on the details being filled in at a later stage.

As noted by Lerdorf and Tatroe (2002, p.141), debugging and maintenance of programs is much easier owing to encapsulation. Debugging is easier because one knows where to look for bugs since the code that change an object's data structures are in the class. Maintenance is easier because one can swap out implementations of a class without changing the code that uses the class, as long as one maintains the same interface. The following will be illustrated to students:

- How to declare properties / attributes and specify the scope of these attributes
- How to hide specific members of an object: What are the visibility scopes available?
- How to access encapsulated data and methods?

3.4.5 Inheritance

Inheritance is a way of defining a new class using an existing class, with certain changed or new properties and methods, thereby serving as a form of code reuse (Lerdorf & Tatroe, 2002, p.141). The new class, called the derived class, inherits the attributes and behaviours of the original class, called the base class or super class (Zak, 2001, p.421). As further stated by Lerdorf and Tatroe (2002, p.141), any improvements or modifications to the base class are automatically passed on to the derived class. The following will be illustrated to students:

- How to inherit the properties and methods from another class?
- How to obtain the inheritance tree for an object?
- How to abstract certain methods: How to override the abstracted features in derived classes?

3.4.6 Polymorphism

Polymorphism is another important concept of object-oriented programming. According to Sebesta (1996, p.354), a polymorphic subprogram is one that takes parameters of different types on different activations. Deitel and Deitel define polymorphism as follows:

> The ability for objects related by inheritance to respond differently to the same message (i.e., member function call). The same message sent to many different types of objects takes on 'many forms' – hence the term polymorphism. (Deitel & Deitel, 1998, p.566)

As further noted by Deitel and Deitel (1998, p.567), polymorphism promotes extensibility; software is written independently of the types of object to which messages are sent. Thus, new types of objects that can respond to existing messages can be added into such a system without modifying the base system. Except for the "client-code" that instantiate new objects, the programs need not be recompiled. The following will be illustrated to students:

- The dependence of polymorphism on inheritance.
- How to implement polymorphism?

Given that the object-oriented analysis-design-and-programming paradigm is a new concept, the above will be taught for three weeks, ten thirty-minute periods per week, at the end of which students should be able to show an understanding of the contents demonstrated by successfully answering questions such as the one given below:

3.4.7 Question

Question 3.4.7.1: Inheritance makes it possible to replace existing methods (which we inherit from the superclass) with ones that are more suited to the derived class. Define a class to represent a square with rectangle as a subclass. Square has one attribute, length. Rectangle provides one attribute, breadth, and inherits length from Square. With geometric shapes such as these, it is often useful to provide methods such as area and perimeter. Write an object-oriented Web-based program that will calculate the area and perimeter for both Square and Rectangle as follows:

Area of Square = 2 x length
*Area of Rectangle = length * breadth*
Perimeter of Square = 4 x length

*Area of Rectangle = 2 * (length + breadth)*

The methods for both Square and Rectangle should have the same name i.e. area and perimeter (Garside & Mariani, 2003, p.300-303).

3.5 Web techniques

Web application development is primarily about creating dynamic Web applications. However, dynamic contents are usually surrounded by static contents. Therefore, the generation of both static and dynamic contents has to be taught. This involves topics such as Web page serving, HTTP Basics, Server information, Form processing, maintaining *Session* and *Application* states.

3.5.1 Web page serving

This involves the creation, loading and redirection of Web pages. Students will be taught the following:

- How to install "normal HTML code" on the Web server? This involves showing which directory the "HTML files" will be stored in so that a Web page will be displayed on typing the server URL address plus the file name. Students will be shown how the appropriate folders/directories are mapped relative to the default Web root directory.
- How to load the installed Web pages from the browsers? This involves entering appropriate URLs that are mapped to the installed HTML files in relative folders.
- How to redirect Web pages to other pages? This will involve a revision of how pages are linked in HTML and how Web forms call the necessary programs to preprocess the form data.

3.5.2 HTTP basics

The Web runs on HTTP, the Hyper Text Transfer Protocol, which governs how Web browsers request files from Web servers and how the servers send the files back. To understand the various techniques in Web application development, one must have a basic understanding of HTTP (Lerdorf & Tatroe, 2002, p.158). Therefore, students will be taught the following:

- What happens when a browser requests a Web page? This involves knowledge of the HTTP request message that is sent to the Web server; its content which includes some header information, and sometimes a body; the Web server's response message which also includes some header information and usually a body (Lerdorf & Tatroe, 2002, p.158).
- The internal components of the HTTP request: This includes knowledge of;

  ➔the first line of an HTTP request which looks like:

  GET /index.html HTTP/1.1

  and specifies an HTTP command, called a method, followed by the address of a document and the version of the HTTP protocol being used.

  ➔the next lines which may contain optional header information that gives that server additional data about the request, such as information about the browser and the MIME types that the browser accepts.

  ➔The blank line that indicates the end of the header section

  (Lerdorf & Tatroe, 2002, p.158).

- The internal components of the HTTP response sent by the server: According to Lerdorf and Tatroe (2002, p.158), this includes knowledge of;

  ➔the first line of the HTTP response which looks like:

  HTTP/1.1 200 OK

  specifying the HTTP protocol version, a status code such as "200" meaning that the request was successful, hence the description OK.

  ➔The response header lines that provide the client with additional information about the response such as the date and time of the response, the Web server software, the type of data (text/html), content length *etc*.

  ➔The blank line that is meant to separate the body from the requested data, which follows if the request was successful.

3.5.3 Server information

Much information about the Web server and the client will usually be made available to the Web applications. This is necessary for effective Web management. Although some

server information are obtainable from the response header, students will be taught how further information can be obtained from the server. Such further information, according to Lerdorf and Tatroe (2002, pp.160-161) includes:

- The hostname or IP address of the host computer.
- The server port number to which the request was sent.
- The method the client used to fetch the document (e.g. GET).
- The path information showing extra path elements provided by the client (e.g. "/students/science faculty".
- The value of the path information, translated by the server into a filename (e.g. "home/httpd/htdocs/students/science_faculty").
- The script name containing the URL path to the current page, which is useful for self-referencing scripts.
- The query string containing everything after the "?" in the URL (e.g., "name=Johnson+age=38").
- The name of the machine that requested this page (e.g., "johnson.tng.ac.za").
- The IP address of the machine that requested this page (e.g., "196.20.220.58").
- The authentication method that was used to protect the page (e.g., "basic"), if the page is password-protected.

3.5.4 Form processing

Using *form data* is the most usual way of moving data from a Web page to a server-side program (Hall, 1999). Extracting the needed information from this form data is traditionally one of the most obvious parts of Web programming that students will first experience. Students will be taught many tricks and techniques for working with forms.

- There are two major ways in which the information can be sent from the Web form to the Web application. It can be attached to the end of the URL after a question mark, for `GET` requests, or sent to the server on a separate line, for `POST` requests (Sun Microsystems, 1997). The significance of using POST rather than GET, with respect to the security of the transmitted data, needs to be pointed out to students.

- The techniques of implementing self-processing pages that both generate the form and process it will also be taught. The presence or absence of parameter values in the form can be used to indicate what to do (Lerdorf & Tatroe, 2002, p.166). Also, hidden fields in the form can be used to indicate whether the form has been processed once, allowing the decision to skip certain parts of the script (Hall, 1999).
- Methods of extracting the sent information from *form data* differs for various Web applications platforms. This must be clearly indicated to students, along with the ways of converting one type of data to another.
- File Uploading is becoming increasingly common in Web applications. Students will therefore be taught how to handle file uploads.

3.5.5 Form Validation

When data are entered into any program, it is necessary to validate the data before using it or storing it for later use. There are several strategies available for validating data. The first is JavaScript or VBScript on the client side. However, since the user can choose to turn JavaScript or VBScript off, or may even be using a browser that does not support it, this should not be the only validation that one effects (Lerdorf & Tatroe, 2002, p.173). A more secure choice is to use the dynamic Web platform to do the validation.

Students will be taught how to do the following:

- Check that a value was supplied: i.e. form variables are non-empty.
- Check that an email address is valid by requiring an "@" sign, checking for the presence of period "." in the domain name, checking the last two or three digits of the domain name against the country code and organization domains respectively.
- Check that a supplied filename is local and exists. This can be ensured by allowing the "browsing" of the filename on the required media.
- Validate a field to ensure that it contains a nonnegative integer.
- Ensure correctness of the comparison of a form value with other values by trimming leading/trailing spaces, converting to lowercase/uppercase *etc*.

3.5.6 Maintaining Session and Application states

HTTP is a stateless protocol (Lerdorf & Tatroe, 2002, p.173; Sun Microsystems, 2002, Module 8: p.3), which means that once a Web server completes a client's request for a Web page, the connection between the two ceases to exist. In other words, there is no way for a server to recognize that a sequence of requests all originate from the same client.

State is useful though. To build a shopping cart application for example, it is necessary to keep track of a sequence of requests from a single user (Lerdorf & Tatroe, 2002, p.178). Once session variables are set, they can be accessed without having to pass them between pages explicitly (Westman, 2002). According to Hall (1999), maintaining states can help the server to provide visitors with a number of conveniences:

Firstly, it enables identifying a user during an e-commerce session. Many on-line stores use a "shopping cart" metaphor in which the user selects and adds items to his shopping cart, and then continues shopping. Since the HTTP connection is closed after each page is sent, when the user selects a new item for his cart, how does the store know that he is the same user who put the previous item in his cart? Maintaining states using a variety of techniques is a good way of accomplishing this.

Secondly, it avoids username and password. It is inconvenient to remember the username and password for many sites. A good alternative is for the server to give a unique user-ID, and when the client reconnects at a later date, the user-ID is returned, and the server looks it up and does not require an explicit username and password.

Thirdly, it allows customizing a site. Many "portal" sites allow one to customize the look of the main page. State maintaining techniques can be used to remember what was wanted, so that one obtains the same result next time.

Lastly, it enables the focusing of advertisement. The search engines are likely to charge their customers much more for displaying "directed" advertisements than "random" ones. This is possible if the server "remembers" clients' interests.

Students will be exposed to the many techniques available to get around the Web's lack of state, and to keep track of state information between requests (also known as session tracking). These include using Hidden form fields, using *Cookies*, and URL rewriting.

- Using Hidden form fields: One such technique for session tracking is to use hidden form fields to pass around information (Lerdorf & Tatroe, 2002, p.178). A more common technique is to assign each user a unique identifier and pass the identifier around using a single hidden form field. However, as noted by Lerdorf and Tatroe (2002, p.178) and Sun Microsystems (1998), hidden forms only work for a sequence of dynamically generated forms, so they are not as generally useful as some other techniques.
- URL rewriting: Another technique is URL rewriting, where every local URL on which the user might click is dynamically modified to include extra information (Lerdorf & Tatroe, 2002, p.178; Sun Microsystems, 2002, Module 8, p.21) such as a user-identification or session ID. Sun Microsystems (2002, Module 8: p.22) states, however, that the URL-rewriting strategy is not as transparent as the *Cookie* strategy.
- Using *Cookies*: A third technique for maintaining state is to use *cookies*. A *cookie* is a bit of information that the server can give to the client. On every subsequent request, the client will give that information back to the server, thus identifying itself (Hall, 1999: Lerdorf & Tatroe, 2002, p.179).

The main problem when using cookies is that some browsers do not support them, and even when they do, the user can disable *cookies*. Other problems include security threats, which Hall (1999) regards as misinformation.

Hall (1999) further remarks that even though they do not present a serious *security* threat, cookies can present a significant threat to *privacy*. Moreover, Hall (1999) indicates that there are still a number of relatively tedious details that need to be handled. These include extracting the cookie that stores the *Session* identifier from the other *cookies* (there may be many, after all) and setting an appropriate expiration time for that *cookie*.

Using *cookies*, students will be taught the following:

- How to retrieve the *Session* object from the request object passed?
- How to store the *Session* attributes?
- Accessing the *Session* attributes.
- Destroying the *Session*.
- Tedious details, problems and limitations of *Session* management.

With regard to the *Application* states, the *Application* object stores state information for use by script files and can also be used to share information among all users of a given application. One can therefore use *Application* variables to store global information, such that different users can modify *Application* variables simultaneously (Microsoft Corporation, 2002, Module 5, p.18). A common use of application state is to record and maintain the number of times a Web page has been visited. The following will be taught:

- How to create the *Application* objects?
- How to lock the *Application* objects?
- How to store the *Application* objects?
- How to unlock the *Application* objects?

The above will be taught for three weeks, in ten thirty-minute periods per week, at the end of which students should be able to show an understanding of the contents demonstrated by successfully answering questions such as the one given below:

3.5.7 Question

Question 3.5.7.1:

a) *Create the Web page in figure 3.2 below.*
b) *Validate the surname field to ensure that it must always be filled.*
c) *Validate the test marks fields to ensure that marks entered are between 0 and 99.*

On clicking "Submit", your page loads another page that welcomes the user with his / her title e.g.

Hello, Dr/ Mrs Dehinbo. Welcome to the world of Web programming.

The title must be assigned such that it can be displayed in subsequent pages. It is assigned as follows:

Title is a "two-word" title if the highest educational title is "Prof." or "Dr". It consists of the educational title along with the assessment of the marital status and gender as

given below. For example, "Dr Mrs" is assigned to a Female Married person with "Dr" as highest educational title. An exception to this is the case of "Prof Mr" which is simply put as "Prof."

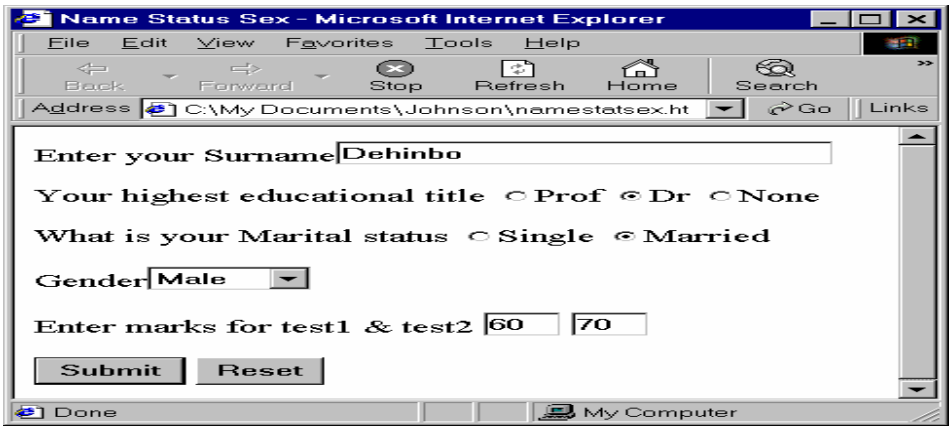

Figure 3-2. Question illustrating Session and Application state management

When there is no high educational title, then the title is a one-word title assigned as follows:

- ☑ *If "Single" and "male", assign the title "Master".*
- ☑ *If "Single" and "Female", assign the title "Miss".*
- ☑ *If "Married" and "Female", assign the title "Mrs" and*
- ☑ *if "Married" and "Male", assign the title "Mr".*

Then, on the next page, display all the particulars of the user including the assigned title and the average of the two tests entered.

Include a link to a third page using the hypertext Click to go to bye- bye page. This page monitors the number of accesses to the Web page and display:
Thanks "title" "Surname". This is the "1st" access to my Web application. e.g.
Thanks, Mr. Dehinbo, this is the "20th" access to my Web application.
For the first, second and third access, it uses 1st, 2nd and 3rd respectively. For others, simply use the number of access with "th" e.g. "24th".
***Note:** The "title" and "Surname" values are those assigned in previous pages.*

3.6 Databases: Remote storage, retrieval and management

Databases are the power behind dynamic Web page generation. Adida (1997b) states that good websites are database-backed though not all database-backed websites are good. Dynamic Web page generation allows the user to connect to up-to-the-minute data, search it and display it in different ways (Yerkey, 2001). Moreover, where Web contents

changes often, storing the data contents in a database allows for efficient searching of the content, and therefore for a stable, flexible and scalable website one needs the database advantage (Adida, 1997b).

3.6.1 Relational databases

The most common type of database in use today is the Relational Data Database Management System (RDBMS) which is a server that manages data. The data is structured into tables, where each table has a number of columns, each of which has a name and a type (Lerdorf & Tatroe, 2002, pp.190-191). Students will have to review:

- How to create database and tables?
- How to add and remove field names?

3.6.2 SQL Syntax

Dynamic Web platforms communicate with relational databases using the Structured Query Language (SQL) to create, modify and query relational databases. The syntax for SQL is divided into Data Manipulation Language (DML) and Data Definition Language (DDL). DDL is the set of SQL commands used to create and modify the database structures that hold the data. DML is used to retrieve and modify data in an existing database, and is remarkably compact, consisting of only four verbs: select, insert, update and delete (Lerdorf & Tatroe, 2002, p.191). Students will also have to review:

- How to use SQL statements to create database and tables?
- How to use SQL statements to insert, retrieve and delete data?

3.6.3 Use of Application Programming Interfaces

Dynamic Web platforms support varying numbers of DBMSs which are usually accessed via Application Programming Interfaces (APIs) such as JDBC (not Java Database Connectivity) (Sun MicroSystems, 1997), Object Linking and Embedding Database (OLE DB) as well as Open Database Connectivity (ODBC) (Microsoft Corporation, 2002, Module 8, p.19). According to Sun MicroSystems (2001, Module 4, p.2), the interfaces serve as a layer of abstraction that allows writing database applications without being concerned with the underlying details of a particular database.

Students will be exposed to these database access interfaces that provide access to a variety of data and information sources which work with various tools and languages and platforms. From the author's experience, the above will involve showing the students:

- How to set up (or specify) the drivers (if necessary) that control the access?
- How to establish a connection with the database file on specified location on remote disks; opening the database connection?
- Retrieving values of the database fields; issuing and executing SQL queries that will retrieve the data values?
- How to extract the data returned by the query, and how to temporarily store the retrieved values into arrays or datasets for easy manipulation of the data?
- Knowledge on navigating the records; how to extract the first record, the records that follow, as well as how to determine the last record in the database.
- Storing values of the form onto the database fields on file; issuing and executing SQL queries that will store the data values.
- How to close the database connection as well as the need for closing the database connection at the end of the database operations?

Apart from remote database storage and retrieval, common operations that are performed on databases are the modification (update) and deletion of database records. Change is a constant phenomenon in life. Names, addresses and other attributes of individuals change frequently. With the advent of Web applications, these changes must be effected as soon as possible in order to avoid consistency and integrity problems.

The above will be taught for three weeks, in ten thirty-minute periods per week, at the end of which students should be able to show an understanding of the contents demonstrated by successfully answering questions such as the one given below:

3.6.4 Questions

Question 3.6.4.1: Create a database in Microsoft Access with the following structure:

studnumber	*integer*
surname	*text*

initials	*text*
sex	*text*
diploma	*text*
subject1	*integer*
subject2	*integer*
subject3	*integer*
subject4	*integer*
subject5	*integer*
subject6	*integer*

Then add at least two records to the database table, and write Web application program(s) to access and display all the stored records in the database. The program should also give the time when the program begins execution and the time after the last record is displayed. The outputs from the program should look like the page given below in figure 3.3:

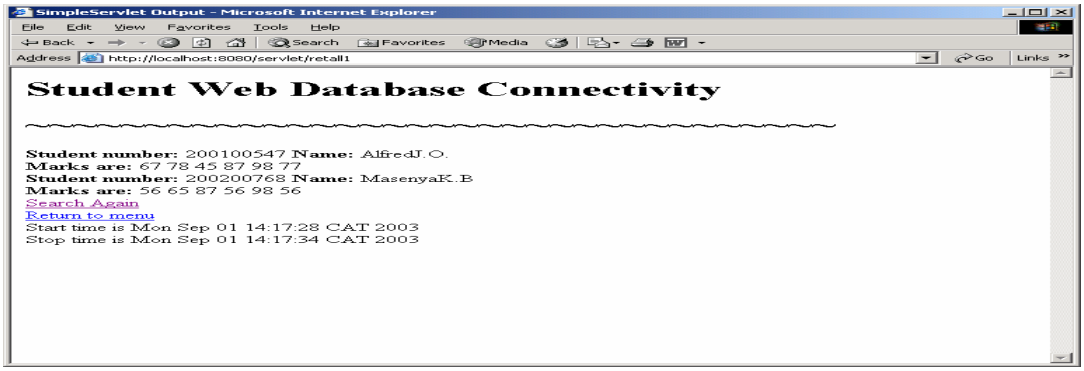

Figure 3-3. Remote Database records retrieval page

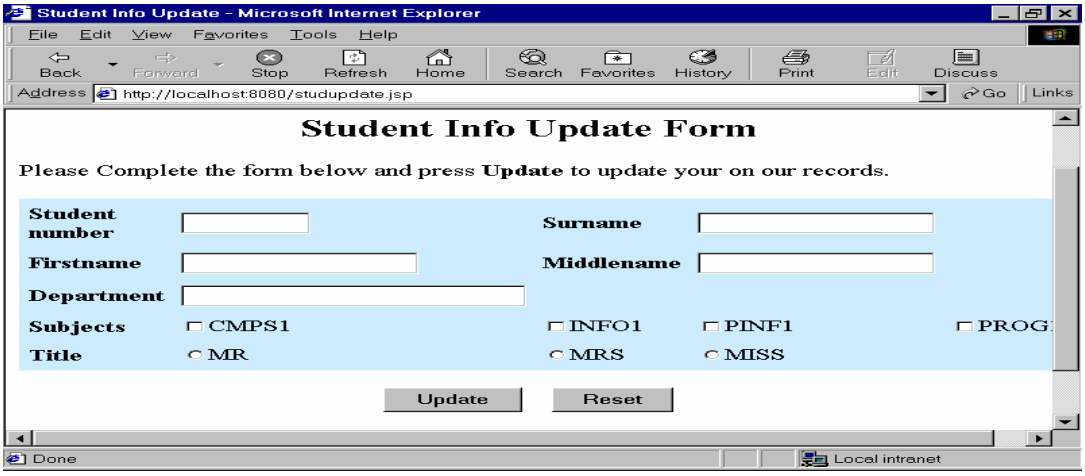

Figure 3-4. Student information update page

Question 3.6.5.2: Remote update of data in relational database: Write a Web application that can be used to accept either the surname or the student number of a student, in order to fetch and modify the whole record for that student. The record is then displayed using the format in figure 3.4 filled with the values for the text fields:

The user can then modify the page to contain the desired information for the five text fields, while new values for the subject and title filled can be clicked, after which the Update button is clicked, and the program will now effect the modifications on the database. Then an acknowledgement page is displayed.

3.7 File Processing

Apart from the remote storage of data onto databases, it may sometimes, be necessary to store remote data on files in the American Standard Code for Information Interchange (ASCII) / text format, or to retrieve such plain text data from a remote file. This could be for performance reasons.

According to Visser (2003), databases are specially built for data storage and have built-in features for security, locking, report generation, relationships etc. Though these features are very helpful, Visser (2003) points out that they also slow down the performance of data storage and retrieval. Therefore, especially in situations where we are only interested in storing data without any need for special features, it is more efficient to store directly on ASCII text files.

3.7.1 File processing techniques

Given the above, students must be aware of methods of storing directly onto files in addition to database storage. Thus, the following will be addressed:

- How to create text files with timestamps such that they do not overwrite one another?
- Storing data from Web forms onto the text files.
- Retrieving data from the text files.
- Uploading documents in files.

The technique for the above will be taught for one week, ten thirty-minute periods per week, at the end of which students should be able to show an understanding of the contents demonstrated by successfully answering questions such as given below:

3.7.2 Question

Question 3.7.2.1: Create a Web application to accept the following data as in figure 3.5 below. On clicking the "Submit" button, your program should store the responses in a remote text file on disk on the server, which in this case will be the lecturer's computer.

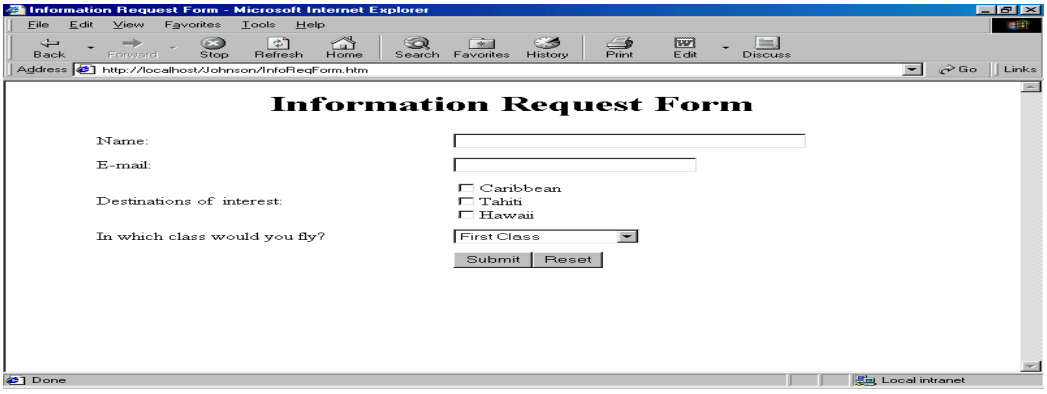

Figure 3-5. User information acceptance page for transmission onto text file

3.8 Exception handling in Web applications

One of the main difficulties of writing a program is making sure that it is robust. That is, when something unexpected happens (e.g. an invalid piece of data has been typed in by the user or an access to the Internet has failed), we have to ensure that the program can detect what has happened and do something about it (Garside & Mariani, 2003, p.351). The ability of a program to intercept run-time errors (as well as other unusual conditions detected by the program), take corrective measures and then continue is a great aid to reliability (Sebesta, 1996, p.17). This language feature is called Exception Handling.

According to Farrell (2003, p.542), an exception is an error or unexpected condition. The program can generate many types of potential exceptions, such as when a command is issued to read a file from a disk, but the file does not exist there or data items are to be written to a disk, but the disk is full or unformatted. Other sources of exception includes

when a program attempts to divide a value by zero, access an array with an invalid subscript, or calculate a value that is too large for the answer's variable type.

These errors are called exceptions because, presumably, they are not usual occurrences; they are "exceptional". A reliable language should be able to handle these exceptions adequately. The next section looks at different types of errors.

3.8.1 HTTP errors

A common type of error in Web applications is the class of HTTP errors. According to Sun Microsystems (2002: Module 9, p.3), an HTTP response sent from the Web server to browsers includes a status code which indicates whether the request was successful (the 200 status code) or not. When unsuccessful, the status code in the 400-500 range indicate the errors types.

By default, the Web browser generates a message based on the status code, and displays it as an HTML message to the user. This is a generic HTTP Error page (Sun Microsystems, 2002: Module 9, p.4). However, as noted by Sun Microsystems (2002, Module 9, p.7), the generic HTTP Error pages generated by the Web browsers (for HTTP error code) and the Web container (for the programming exceptions) are often ugly and not very informative to the user. Sometimes, they are even frightening. Students will therefore be taught:

- How to extract the status code from the HTTP response.
- How to create new error pages with informative messages and allow the Web container to handle the forwarding to these pages.

3.8.2 Other programming errors

In addition to HTTP errors, a Web application can generate exceptions to indicate a problem with the processing of the HTTP request. Students will be exposed to various programming techniques to handle such errors. These include:

- How to intercept and handle run-time errors.

- How to specify the errors that the program is allowed to leave unhandled.
- How to exit gracefully without stopping the program execution.

3.8.3 Logging exceptions

In addition to handling errors interactively, an exception might be written to a log file. This is necessary because, given that the standard screen is limited to about 25 lines per display, the errors sometimes scroll up, leaving the user with the last set of errors. Moreover, programmers are often very tense when debugging. Therefore, when the exception is written to a log file, the programmer can look at it later in a more relaxed mood. Students will be taught:

- How to create a file on the server from the executing Web application.
- How to log exceptions to the created file.

The techniques for the above will be taught for one week, ten thirty-minute periods per week, at the end of which students should show an understanding of the contents demonstrated by successfully answering questions such as those given below:

3.8.4 Questions

Question 3.8.4.1: Write a Web application that intentionally contains some errors such as "non-existing database file". Then let your application intercept the error and display user-friendly messages about the error in a Web page to which the system is directed by the Web browser, and then continue operations. Also let your application write the error into a log file.

Question 3.8.4.2: Write a Web application that intentionally contains some syntax errors such as "misspelled command name". Then write exception handling code that will intercept the error and display user-friendly messages about the error, and then continue.

3.9 XML support

According to Lim (2002, p.4), a new component that has to be added to the current offering of a Web development course is the ubiquitous XML, because it has "mushroomed" so quickly in the industry that a Web development course without XML is just not complete. XML stands for eXtensible Markup Language, and XML 1.0 is a

standard of the World Wide Web Consortium (W3C)'s recommendations (Sun Microsystems, 2002, Appendix E, p.3). A good description of XML is given below:

> XML is a markup language that provides a format for describing structured data. XML enables precise declarations of content and provides a structured representation of data that is easy to implement (Microsoft Corporation, 2000, Module 8, p.12).

Most XML documents consist of elements (HTML tags, entities, and regular data). The term "extensible" is derived from the fact that, unlike HTML with basic tags with predefined meanings, XML allows one to create unique tags that are meaningful to one's data (Lerdorf & Tatroe, 2002, p. 262; Sun Microsystems, 2002: Appendix E, p.3). XML is not a technology but a standard, and as such, other tools and technologies make use of the XML standard for a uniform way of transferring and manipulating data (Botha & Eloff, 2002).

Lerdorf and Tatroe (2002, p.262) point out that as a common format for data transfer, other programs can emit XML files for one to either extract information from (parse) or display in HTML (transform). Coupled with XML's recent use in Remote Procedure Calls (RPC), XML has proved a useful way of integrating application components written in different languages.

Due to the importance and relevance of XML as highlighted above, students will be exposed to knowledge and practice on XML. Below is a detailed view of what will be taught to students:

- How to define both the content and the structure of the data?
- How to separate or decouple the structure of the data from the presentation of the data? (Bertino & Catania, 2001).
- How to ensure that the structure of the data is open and extensible?
- Understanding the parts of an XML document would include the processing instructions, elements, child elements, attributes and comments (Microsoft Corporation, 2002, Module 12, p.3).
- Understanding the various rules for defining the elements.

- Understanding the various major similarities and differences between XML and HTML. An example is XML's extensibility that allows one to create unique tags that are meaningful to one's data (Lerdorf & Tatroe, 2002, p.262; Sun Microsystems, 2002, Appendix E, p.3), e.g. <authors> tag which is not predefined as in HTML.
- How to embed XML data in HTML or reference the XML data from external file?
- How to attach a style sheet to the XML data to display the data appropriately?
- How to transmit XML document from a program?
- How to take in an XML document and parse the data contained?

Given the importance of XML support as highlighted so far, the techniques for the above will be taught for one week, ten thirty-minute periods. At the end of this period, students should be able to show an understanding of the contents demonstrated by successfully answering questions such as those given below.

3.9.1 Questions

Question 3.9.1.1: Create an HTML Web page and embed an XML data island in the HTML page, and bind the elements to an HTML table. The XML data island should contain at least four employee records containing three fields, namely the First name, Last Name and Designation.

Question 3.9.1.2: Develop a Web program to fetch all records from a database file and emit these records as an XML document.

3.10 Other desired constraints to be satisfied

In addition to ensuring suitability for teaching desired Web application development concepts, it is important that a platform selected using our proposed framework should satisfy other desired constraints. These include ensuring that the chosen platform is easy to use for second year students, and fast to use so as to satisfy the recurrent editing-compilation/interpretation-testing that takes place within the limited period of the practical class. These are necessary given that second year students who will be using the platforms are still at the beginners' level in application development.

Other constraints include the affordability and portability of the platforms. The discussion of these constraints in the next section will enable us to develop various criteria and experiment to measure the satisfaction of the constraints in the framework.

3.10.1 Usability, ease of use and ease of learning

Usability is a key concept in Human-Computer Interaction. From the perspective of users, designed systems such as websites and Web applications must be completely usable or people will not re-visit the website. In the article *Centers of achievement* in the Science and Technology Advertorial column of Sunday World on 2 November 2003, "Usability" is explained as referring to how easily people learn to use computer programs, the Internet, cell-phones or even power tools (TelkomInternet, 2003). According to Preece *et al.* (2002) cited in Singh (2002), usability is generally regarded as ensuring that interactive applications are easy to learn, effective to use, enjoyable, and involve the optimization of user interaction with these products.

The framework being developed in this study focuses on determining the suitability of dynamic Web platforms for teaching Web applications development. This is closely related to the concept of usability except that here we are not considering the usability of a designed software product. Rather, we are interested in evaluating how usable the platforms used to develop the Web applications are. We have therefore decided to extend the meaning of the term "usability" in this study to mean "ease of use and ease of learning of the platforms".

According to Ashenfelter (1999, p.111), Web databases require a knowledge that spans the worlds of Webmaster, database administrator, and programmer. Therefore, learning to use dynamic Web platforms is relatively difficult compared to other programming languages. Ashenfelter (1999, p.112) notes that the ease with which new tools can be learned has an impact on whether and how much they will be used.

Ease of learning in this context is a measure of how easy it is to learn the use of a particular dynamic Web platform in order to develop Web applications using it.

Ashenfelter (1999, p.112) identifies a number of factors that can affect how easily a Web platform can be learned. Firstly, unnecessarily complex platforms are the biggest obstacle to ease of learning. Secondly, familiarity of the related components of the dynamic Web platform will enhance learning. Thirdly, availability of useful documentation will be of great assistance in learning the platform. Lastly, the cost of the platform and the cost of training will also affect its learning.

Ashenfelter (1999, p.113) indicates that the fact that a platform is easy to learn does not mean it will be easy to use, and vice versa. A platform that requires little coding to achieve a particular effect will be easier to use than an easy-to-learn one that requires a great deal of work to produce required effects.

In this context "ease of use" is a measure of how easy it is to use a particular Web platform to develop dynamic Web applications. This is important given the need develop Web applications using a particular dynamic Web platform in urgent circumstances, such as during a practical class when time is limited or in an examination situation. Measuring the ease of use by our framework will ensure that the selected platform is easy to use for second year students of Web application development.

Closely related to ease of use is the performance of the platform. As highlighted in the introductory chapter, a platform that is very slow while executing developed programs will make program testing difficult. The next section therefore examines performance of platforms and related concepts such as latency and throughput.

3.10.2 Performance, latency and throughput

The term "performance" as used in this study refers to the total time in which the operations within the Web applications are executed. For many systems, poor performance is often an inconvenience and perhaps a source of complaints, but the users keep using the system. But in Web and e-commerce systems, performance problems are much more than an inconvenience – they can be a disaster for a business, putting off

customers and giving competitors an advantage (Treese & Stewart, 2002, p.177). So developing Web applications that are fast, reliable and scalable is one of the most challenging parts of creating Web and e-commerce systems (Treese & Stewart, 2002, p.173).

According to Marshak and Levy (2003, p.1), the central performance problem in the World Wide Web in recent years has been user perceived latency. Treese and Stewart (2003, p.177) state that the performance of a system can be measured in many ways, including using metrics that examine latency and throughput. Latency is a measure of how long it takes to complete a given operation, e.g. how long it takes to download a Web page. Throughput is a measure of how many operations can be completed in a given time, e.g. how many Web pages the server can deliver (transactions) per second.

In their comparison of latency and throughput, Treese and Stewart (2003, p.177) went on to add that while latency informs one of the experience of a particular user (on average), throughput tells one how many users the system can handle. As throughput increases, for example, the latency as seen by any given user may increase[11]. It is thus enough to measure either latency or throughput. An experiment will be established as part of our framework, to measure latency of Web applications using various selected platforms.

3.10.3 Performance and scaling

According to Treese and Stewart (2003, p.176), scaling is the question of how large a system can grow in various dimensions to provide more service. It can be measured by the total number of users, the number of simultaneous users, transaction volume and so on. Scaling in one dimension typically affects other dimensions, e.g. increasing the size of a database to handle more users may sometimes decrease the performance (Treese & Stewart, 2003, p.176). We will therefore incorporate scaling into the latency experiment being developed as part of our framework.

[11] As illustrated by Treese and Stewart, this is like driving in a heavy traffic; the number of cars moving down the highway is greater than normal (i.e. greater throughput) but the average speed is lower, yielding higher latency (Treese & Stewart, 2003, p.117).

3.10.4 Portability, Compatibility and Cost

Compatibility of a platform with various databases, servers and browsers as well as compatibility with open standards and other international standards is of paramount importance (Ashenfelter, 1999, p.123). Open standards and propriety solutions such as ODBC and SQL enhance the ease of working with different databases. Of great importance is the portability across major operating systems including Windows, Unix and MacOS (Ashenfelter, 1999, p.133).

Portability can be influenced by many other factors such as the architectural neutrality, applicability to a wide range of applications and the completeness and precision of the language's official defining document, the programming environment (Sebesta, 1996, p.19). Architectural neutrality aims at freeing languages from implementation details (Wigglesworth, 2000, p.8), while portability is the ease with which programs can be moved from one implementation to another (Sebesta, 1996, p.19).

Compatibility and portability can be linked to cost. This is because most open-source platforms are compatible with open standards and are usually portable. Examples include Java and PHP.

An influential factor in the choice of a platform is the cost of executing programs written in a platform as well as the cost of acquiring the platform, its Web server and related software such as the database management systems and the IDE. If a particular cost in a platform is high, there will be no interest in moving any previous work to such an implementation platform, even if it is technically feasible. This is illustrated by the following comment in a discussion article (Fuecks, 2004):

> I was hired because my employer was moving from ASP to PHP. When I asked why they were moving to PHP, my boss said one word: cost.

This becomes even more important when one considers the fact that students have few funds available to purchase software, yet they need to practice in their own time.

Vinoski (2004) asks: "How do you know if the dynamic Web platform you choose will be economically viable as long as you plan to use it?"[12] With the proliferation of various open source software, the cost of acquiring the platform, its Web server and related software will have an effect on the wide usage of the platform. The availability of downloadable materials will also reduce the cost of learning the platform. Therefore, criteria to measure portability, compatibility and cost of dynamic Web platforms will be included in the framework being developed.

3.11 Conclusions

This study aims to develop and apply a conceptual framework that can be used to determine a suitable platforms for teaching Web application development. In order to enable us to identify the criteria that must be part of the framework for evaluating the Web platforms, this chapter discusses the required concepts to be taught to the second year students, as well as the other constraints to be satisfied. These will enable us establish criteria in our framework to ensure the suitability of the platforms.

This chapter is started with a discussion of structured programming concepts that can be used to understand the top-down decomposition of problems into manageable solution components. Closely supporting the structured programming concepts are the object-oriented concepts that are aimed at understanding approaches of modeling the problem domain in terms of real life objects and then building reusable program components. Ebner *et al.* (2000) highlight the need for a high degree of reusability to speed up the application development because Web applications' contents continuously change and so reusability enables isolating and changing relevant components.

A feature distinguishing Web application development from general software development is that applications are meant to execute on the Web. Therefore, this chapter further includes desirable Web techniques that will enhance students' capabilities in

[12] Some people believe that Corba 3.0 is not getting the attention it deserves despite being technically more powerful and advanced than either .NET or J2EE. They claim that this is because Corba vendors charge too much for their products and that their greed has ultimately limited market size (Vinoski, 2004).

developing applications with features corresponding to those available on established Web applications.

In order to be truly dynamic, providing up-to-date information in a real-time interactive form, Web applications store information on databases with built-in features to help in managing the data. This chapter also looks at desired knowledge needed to manage information on the database. This is closely followed by a look at techniques to use text files instead of databases, possibly for performance reasons.

It is important to inculcate in students the attitude of "doing it right". For this reason, apart from satisfying the functional requirements, the Web applications they design must also satisfy reliability constraints by being robust. This chapter thus also looks at the need to design Web applications to handle exceptions by intercepting run-time errors and other unexpected circumstances, and taking preventive, corrective and reflexive measures. Finally, the importance of designing applications that enhance communications with other applications is catered for by including techniques to generate and parse XML documents.

Without doubt, second year students who are equipped with the knowledge of the various concepts discussed in this chapter will be an invaluable asset for any Web application development team, and will be adequately equipped for the future. However, apart from ensuring suitability for teaching the desired concepts, the framework needs to ensure the satisfaction of other constraints such as ease of use, speed of use (low latency), affordability and portability. These are necessary to cushion some of the effects of the complexity of programming so that students can concentrate on problem solving. The last section discusses these factors.

Given the need for adequate knowledge of the above concepts, and the need to satisfy the other necessary constraints highlighted, the next chapter builds on this chapter to establish and develop our framework. This forms the major contribution of this study.

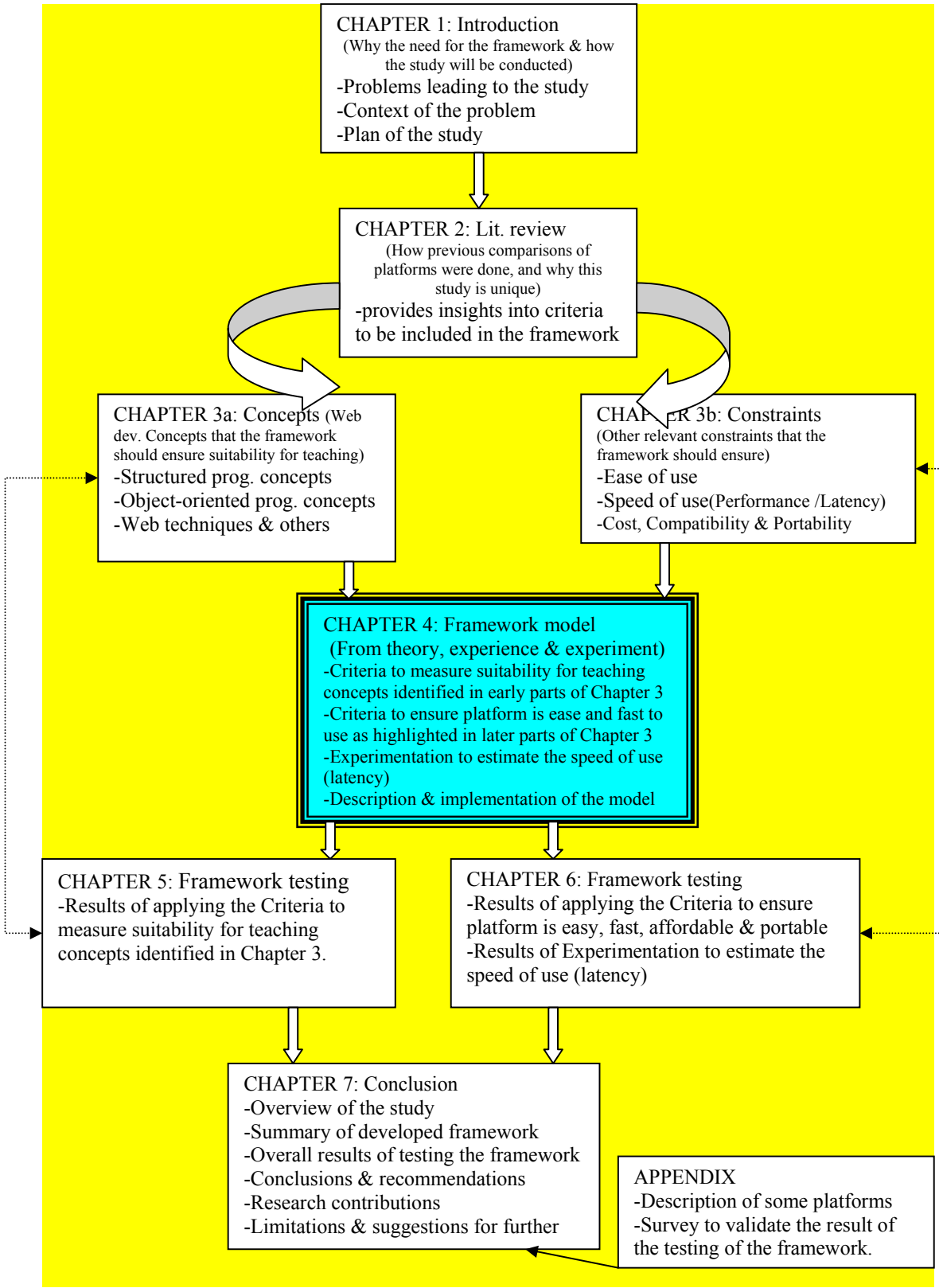

Figure 4-0. Dissertation map highlighting the relativity of chapter 4 to the rest of the dissertation.

CHAPTER 4

4 FRAMEWORK FOR THE EVALUATION OF THE SUITABILITY OF WEB-BASED PLATFORMS

4.1 Introduction

The most important criteria in the framework for determining the suitability of a dynamic Web platform for this study is that it should be useful for the effective teaching of dynamic Web applications development to second year students of tertiary institutions. Inherent in this constraint is that it must build upon the program design concepts taught to the students in their first year of study.

Chapter 3 has established the concepts of Web application development desirable for second year students and the other necessary constraints that should be satisfied. This chapter therefore aims to use chapter 3 to establish the various criteria of the framework to evaluate the suitability of the platforms. This involves identifying qualities that will ease the teaching of the identified concepts, as well as qualities that will ensure that the platform will be easy to use, fast, portable and affordable for second year students.

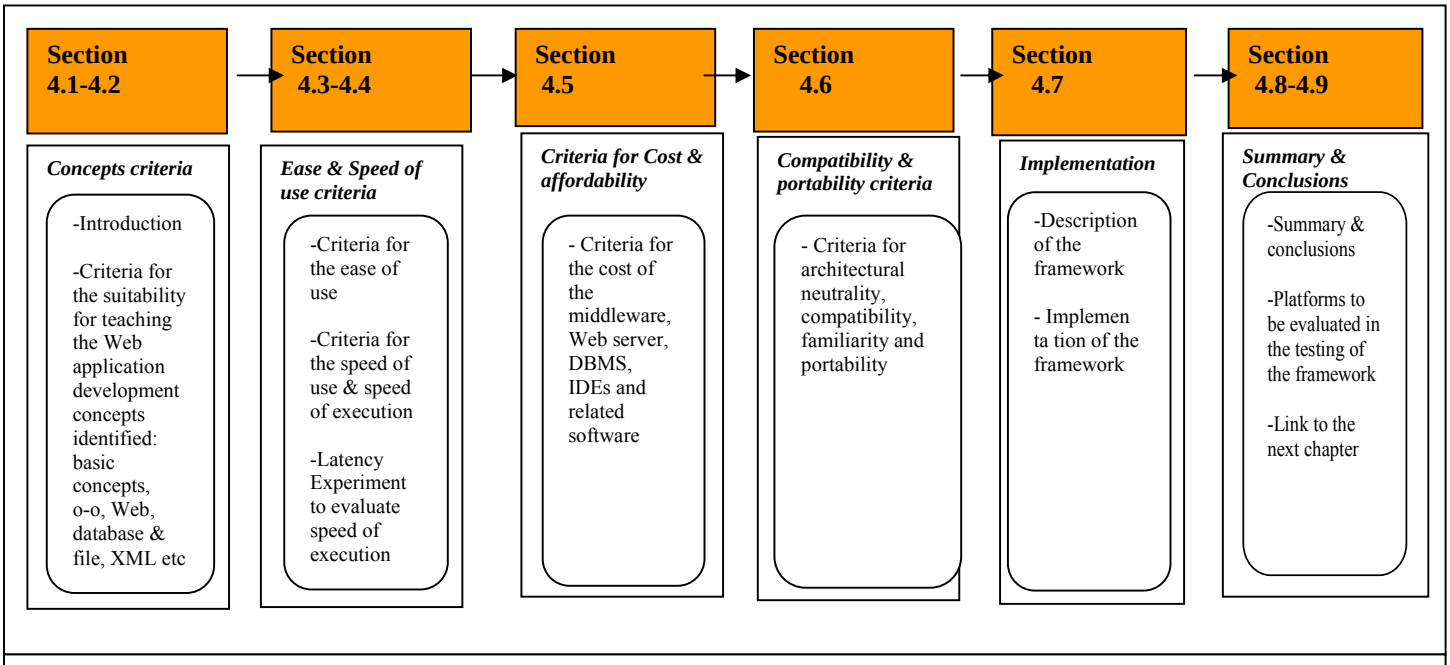

Figure 4-1. Pictorial outline of chapter 4.

This chapter addresses both theoretical criteria and experimental programming criteria. Figure 4.1 gives the outline of the chapter. Section 4.2 lists the criteria that support the suitability for teaching the concepts identified in chapter 3 sections 3.2 through to section 3.9, while section 4.3 establishes the criteria for evaluating the ease and speed of use of the platforms by second year students. The latency experiment is discussed in section 4.4 while the criteria for evaluating the cost and portability are given in section 4.5 and section 4.6 respectively. Section 4.7 presents a description of the framework and the spreadsheet implementation of a tool to use the framework.

4.2 Criteria for evaluating the suitability for teaching the Web application development concepts

It is of primary importance that a suitable platform must be capable of satisfying the purpose for which it is needed, which in this case implies that it must be suitable for the purpose of teaching Web applications development. In this section, we present the desired qualities that will serve as criteria to evaluate the suitability of the platforms for the effective teaching of the Web application development concepts.

4.2.1 Criteria for evaluating the suitability for teaching the basic concepts

This section presents the desired qualities that will serve as criteria for evaluating the effective teaching of the basic concepts and structured programming concepts of Web application development identified in section 3.2 of chapter 3. One of the most important criteria for judging a programming language is the ease with which programs can be read and understood. This is termed readability. Several factors influence readability. These include lexical structure, syntax design, control structures, simplicity, data types and structures, expressivity, and support for abstraction (Sebesta, 1996, p.9).

4.2.1.1 Lexical structure

For a student to be able to write any meaningful program in any language, he needs to have a broad grasp of the lexical structures. This implies that the student should easily recollect the command structures that are to be used in certain situations, as well as the

syntax of these commands. This is even more important in our situation where students undergo timed practical laboratory sessions and practical examinations. It would not be much good for students to spend an enormous amount of time just in identifying that a "For" instead of "for" is the cause of the error in a statement. These types of details will automatically become "part and parcel" of the student as the use of the language progresses. With these in mind, below are the various qualities that will enhance the teaching of the lexical structure of the desired platform:

- ☑ Availability of adopted way or format for forming variable names from compound words, such that programmers can easily remember variable names.
- ☑ Case sensitivity: Highly case sensitive keywords, commands and variable names are good for error checking and reliability but may lead to errors for second year students. Allow students to learn suitable commands and their names first.
- ☑ Statement punctuations: Compulsory punctuation to end statements are minor errors that may not easily be identified. In some languages, the compiler reflects such errors in the next statement, thereby causing many problems for the beginning students who start looking for errors in the wrong place.
- ☑ Multiple statements on a line will make error isolation difficult.
- ☑ Compound statements: Special words being used to mark different compound statements (e.g. if …. endif) will make programming easier for second year students. This is more readable than the use of { and } which becomes confusing when many of them are used to demarcate different commands.
- ☑ Form and meaning: Statements and commands whose appearance at least partially indicates their purpose. Semantics, or meaning, should follow directly from syntax, or form. That will enhance recollection of the commands.
- ☑ Coding structure: Can one go straight to coding without much detailed definition of class and service methods structures as well as output objects? This will allow students to concentrate more on problem solving.

Emerging criteria questions: Lexical structures

1. Are there adopted ways for forming variable names from compound words, such that names can easily be remembered e.g. 'timeOfDay' ?

2. Are the formed variable names highly case insensitive e.g. ‘timeOfDay’ not different from ‘timeOfday’?
3. Are the keywords and commands highly case insensitive e.g. print not different from Print?
4. Is there no compulsory punctuation used to end the statements e.g. no need for using ; to end statements?
5. Are multiple statements disallowed on a line even though separated by special characters e.g. print a*b : calculate c/d?
6. Are there special words used to mark different compound statements (e.g. if … endif) instead of just using { and }?
7. Does the appearance of some statements and commands partially indicate their purpose, meaning or syntax e.g. getString()?
8. Can one spread a statement across any number of lines instead of being forced to have one very long one that is often out of view?
9. Is it required to indicate when the next line is a continuation of the previous line, or is there an easy way of knowing?
10. Can one go straight into coding without much detailed requirement to declare and define class and service method structures as well as output objects?

4.2.1.2 Data types

One of the major steps in solving a problem is the representation of the data items in appropriate data structures. Firstly, the data structure determines some operation possible on the data items, e.g. sorting is made easier with the use of arrays. Secondly, mixing similar data types such as integers and floats may sometimes result in loss of precision, but that should not be the type of error that will prevent the student from executing the small program being experimented with.

As an example, if a student makes such a mistake and sees that instead of getting a result with Rands and cents, only the Rands portion is displayed, such a student will be prompted to make amendments. That is better that being put off by a “type mismatch” error that prohibits further execution. With these in mind, below are the various qualities that will serve as criteria in evaluating the effective teaching of the basic concepts of Web application development with respect to data types:

- ☑ Availability of the *variant data type* support so that a variable can be assigned values of different types: this will enhance the flexibility of data assignments.
- ☑ Support for overflow; errors resulting from overflows are not easily visible to students, especially second year students.

- ☑ Lack of limit for the size of String values: allowing for adequate expression of values for strings, since beginner students will want to express themselves in terms of values being assigned.
- ☑ Ability to extend array size or not declaring a fixed size for arrays: so that students can first try out some techniques with few array elements and then experimentally extend the elements.
- ☑ Ability to "cast" and to mix data types in an expression when necessary: this will reduce occasions of "type mismatch" in intended situations where the error's effect is of little or no consequence to the accuracy of results.
- ☑ Requirement for declaring a variable before it is used in the program: this is necessary so that spelling mistakes in variable names will not be mistaken for a new variable.

Emerging criteria questions: Data types

1. Is there "variant" data type so that a variable can be assigned values of different types, e.g. a variable used for float or string later in same program?
2. Is there support for overflow e.g. are values growing beyond the bound for integer type interpreted as float instead of yielding errors?
3. Is the range for data type values high enough to reduce the situations for overflow?
4. Is there no (or very high) limit for the size of String values that can be assigned to a String variable, such that there is no concern about long strings?
5. Is the size of an array flexible and extendable so that it depends on run time values assigned?
6. Is there flexibility in using either of the parentheses '( )' or '[ ]' in array processing?
7. Are there iterator functions for traversing array elements?
8. Can one cast or mix data types in an expression, when necessary, e.g adding integer and float?
9. Is there flexibility to optionally declare variables or not, before being used in the program?

4.2.1.3 Expressions and operators

In an analogy of children learning to speak simple sentences, appropriate operators need to be used by students to combine with constants, identifiers, and keywords to form simple expressions. The problem domain must be broken down into various simple expressions. Complex expressions will be difficult to comprehend by second year students. Given below are various qualities that may serve as criteria for evaluating the suitability for effective teaching of the ease of decomposing the problem into basic expressions:

- ☑ Abstraction: Availability of simple, relatively convenient, rather than cumbersome ways of specifying computations. For example, the notation *count++* is more convenient and shorter than *count = count + 1*.
- ☑ Operators: Meaningful symbols used for arithmetic, logical and relational operators will enhance recall of the operators.
- ☑ Less operator overloading: Distinct symbols for the different arithmetic and logic operations will enhance readability of the code.

Emerging criteria questions: Expressions and Operators

1. Are there relatively simple and convenient, rather than cumbersome ways of specifying computations, e.g. using the notation count++ instead of count = count + 1?
2. Are the symbols used for arithmetic, logic and relational operators meaningful such that it is easy to remember what the symbol represents?
3. Are the relational keywords highly case in-sensitive, e.g. 'and' same as 'AND'?
4. Is operator overloading discouraged such that the same symbol cannot be redefined for use in different arithmetic and logic operations?
5. Does the division operator always give the same result for Integer division and Floating point division?
6. Is there a separate operator that can be used when only integer division is desired ?
7. Is the use of the same symbol for addition and string concatenation discouraged?
8. Is there any special high level operator for exponentiation?
9. Is the assignment operator different from the equality operator?
10. Is the operator for string comparison the same as the operator for numeric comparison?

4.2.1.4 Control structures

Control structures are analogous to the wheels, gears and steering mechanism of a car. Just as novice drivers quickly learn to drive in an automatic car, so also the availability of appropriate control structures will enable the student to master programming involved in Web application developments. The various qualities that will serve as criteria for the evaluation of effective teaching of the basic concepts of Web application development with respect to the use of control structures are given below:

- ☑ Sufficient control statements such that the need for unconditional branching (such as the "goto" statement in BASIC) has nearly been eliminated: this will enhance readability and desk-checking of the code.
- ☑ Provision for special looping for traversing array elements: the presence of this special loop will clearly indicate array traversing rather than other looping operations, thereby increasing the readability of the code.

- ☑ Possibilities for using both Integers and Strings as array subscripts: while integers have been well used as subscripts, the use of strings will make programs more readable.

Emerging criteria questions: Control structures

1. Are there sufficient control statements such that the need for the unconditional branching (such as the "goto" statement) has been eliminated?
2. Is there a special looping control structure for traversing array elements?
3. Are there possibilities for using both integers and strings as array subscripts?
4. Is the increment in a "for" loop specified in clear wording?
5. Is there a default value for incrementing or decrementing loop control variable?
6. Is the changing of the value of the loop control variable within the loop disallowed?
7. Is the omission of the "break" statement in a multiple selection statement not likely to cause disastrous logical errors of one "case" running into others?
8. Is it possible to use any other type such as "string" as the control variable for the multiple selection statement?
9. Is the control structure block surrounded with meaningful keywords such as if … endif?

4.2.1.5 User-defined functions and modularizations

Effective problem solving requires decomposing the problem into smaller manageable components. Most important for students who have just started learning the concepts of Web application development, are the facilities that will enhance both the decomposition of the problem as well as the building up of Web applications from smaller components. These features should also enhance reuse of code. Therefore, the following qualities will serve as criteria to evaluate effective teaching of user-defined functions and modularizations as part of the basic concepts of Web application development:

- ☑ Provision for prototype definition; defining the prototype will immediately signify the use of functions and also prepare the mind for the expectation of the function calls and function definitions.
- ☑ Support for function overloading; will make it easy to use the same function name to handle different numbers and types of input parameters without extensive use of decision statements that will clutter the whole program.
- ☑ Independent compilation of functional components; will make it easy to build application incrementally from smaller components which can be independently tested.

- ☑ Adequate abstraction using in-built functions: high level abstraction is more comprehensible and more readable.

Emerging criteria questions: Functions and Modularizations

1. Must prototypes for functions be defined before they are referenced later in the program?
2. Is it possible to compile or interpret and execute separate functional components of the program independently?
3. Are there many high level built-in functions, e.g. for sorting, printing etc.?
4. Is function overloading supported such that the same name can be used for the same functions which differ only in type and number of parameters?
5. Are there high level functions for sorting?
6. Is the global variable prevented from overriding local variables?

4.2.2 Criteria for evaluating the suitability for teaching object-oriented concepts

In addition to the support for structured programming capabilities, it is necessary to keep pace with the object-oriented trends in programming languages. For this reason the platform should support the object-based decomposition of the problem domain. Section 3.3 of Chapter 3 lists the object oriented concepts necessary for Web applications programming taught to second year students. This section now presents the desired qualities that will serve as criteria for evaluating the suitability for teaching the object oriented concepts. The detailed criteria are given below:

- ☑ Support for the class definition demonstrating the encapsulation of data with methods: this will enhance the appreciation of the need to group related objects with the encapsulation of their properties and behaviours.
- ☑ Support for constructor functions including overloading of constructors: this will enhance the understanding of the passing of default values to new objects, along with the flexibility of varying the default values for different objects of the same class.
- ☑ Support for information hiding: this is necessary to illustrate how to hide information.
- ☑ Independent compilation of classes: will make it easy to build application incrementally from smaller components which are independently tested and also reusable.

- ☑ Support for classes within classes: this will illustrate the hierarchical nature of classes.
- ☑ Support for inheritance: this is to illustrate the simplicity and impact of establishing inheritance relationships.
- ☑ Support for abstraction with facilities for overriding abstracted features: this is to demonstrate the use of abstraction in solving a given problem.
- ☑ Facilities for checking for the class to which an object belongs: this is necessary to avoid confusion as the number of objects increases.
- ☑ Support for polymorphism: this is to demonstrate the elegance and usefulness of polymorphism.
- ☑ Support for the use of special names such as "parent" to access super-class.

Emerging criteria questions: Object-oriented concepts

1. Is there support for the class definition demonstrating the encapsulation of data with methods?
2. Is there support for constructor functions including the overloading of the constructors?
3. Is there any support for information hiding?
4. Is it possible to independently compile or interpret and execute separate classes so as to build the program incrementally?
5. Is there support for classes within classes illustrating the hierarchical nature of classes?
6. Is there support for inheritance illustrating inheritance relationships?
7. Is there support for abstraction with facilities for overriding abstracted features?
8. Is polymorphism supported?
9. Are there facilities for checking whether an identifier is an object or identifying the class to which an object belongs?
10. Is it possible to use a pseudo-variable like "this" in order to be able to access its own functions and variables from within a class?
11. Is it possible to use special names such as "parent" or "super" to refer to the name of the base class instead of using the literal name in the code?

4.2.3 Criteria for evaluating the suitability for teaching Web techniques

One way in which dynamic Web platforms differ from the general programming languages is in their support for Web development techniques. Section 3.4 of Chapter 3 discusses the Web application development concepts that are used for the generation of both static and dynamic Web contents. This section provides the criteria which will ensure that the Web development features in most Web applications are elegantly achievable with the chosen platform:

- ☑ Facility for static Web page serving by the associated Web server: this is necessary to illustrate the use of the platform for static Web contents.
- ☑ Facilities for extracting information from the HTTP request header: this indicates the use of the request header as a medium for conveying useful information.
- ☑ Facilities to embed or set information into the HTTP response header: this illustrates the use of the response header to convey useful information.
- ☑ Provision of other relevant information by the Web server: this illustrates the wealth of information available through the server.
- ☑ Availability of default folders for placing program code: these will be easier for beginners to use before they have mastered the art of mapping to various directories.
- ☑ Possibilities of sending information via the service method that could implement both *GET* and *POST* methods: this will ensure that the rest of the code are not affected by changes between *GET* and *POST* methods.
- ☑ Facilities for form validation, e.g. checking for non empty fields, range of values, content of fields, and using hidden variables to check if form has been filled once, *etc*.; this will illustrate the importance of form validations.
- ☑ Facilities for identifying users in a session, e.g. using cookies, hidden variables or URL rewriting: this will illustrate the techniques used to overcome the lack of state in the HTTP protocol.
- ☑ Possibility of setting "age" for cookies thereby making cookies last longer or alternatively disallowing cache of pages: this will illustrate techniques for extending the use of cookies.
- ☑ Facilities for maintaining Application objects: this illustrates the techniques for maintaining global Application states.

Emerging criteria questions: Web techniques

1. Are there facilities for static Web page serving by the associated server?
2. Are there facilities for extracting information from the HTTP request headers?
3. Are there facilities for embedding or setting information into the HTTP response headers?
4. Are there ways of obtaining relevant information from the server?
5. Is there availability of default folders for storing program code?

6. Is there the possibility of sending information via the service method that could implement both GET and POST methods?
7. Do numeric values accepted from a Web form retain their numeric type instead of being in string format which still has to be reconverted ?
8. Are there simple facilities or functions for form validation, e.g. checking non-empty fields, range of values, content of fields etc.?
9. Are there facilities for identifying users in a session, e.g. using cookies, hidden variables or URL rewriting?
10. Is there the possibility of setting the age of cookies thereby making cookies last longer?
11. Are there facilities for maintaining application objects so as to maintain global application states?

4.2.4 Criteria for evaluating the suitability for teaching the database handling capabilities

The importance of database handling in a Web application cannot be overemphasized. More especially, the remote handling of the database manipulations forms the hub of most Web applications. Sections 3.5 and 3.6 of Chapter 3 discuss the database handling concepts of Web application development. This section is aimed at establishing the availability of features that will enhance the understanding of remote database manipulations. It is of utmost importance that the communications between the server and the database are achieved with the minimum effort. The criteria below will ensure adequate database handling:

- ☑ Support for various Database Management Systems (DBMS): this will leverage the knowledge of any DBMS the students may already have.
- ☑ Inclusion of specific database in the installation package: this will ease the configuration settings and may be used to test and master the system.
- ☑ Support for common Database Management Systems, e.g. Microsoft Access via both ODBC and JDBC: this will ensure transfer of previous knowledge and also illustrate the use of the two APIs.
- ☑ User-friendly facilities for loading drivers, opening and closing connections, retrieving data items, etc.: this will ensure that the communications between the server and the database are achieved with minimum effort.
- ☑ Possibilities for retrieving items in their original *data types* rather than converting back from strings: this will avoid the error of attempting calculations on String variables or converting to and from Strings.

- ☑ Facilities for navigating the datasets: this will ensure knowledge of looping through the data items fetched.

Emerging criteria questions: Database handling capabilities

1. Does the platform support various database management systems so as to leverage any knowledge of DBMS the students may already have?
2. Is the DBMS sometimes included in the installation package so as to ease the installation configuration settings?
3. Does the platform support MsAccess as the common DBMS via both ODBC and JDBC?
4. Are there user-friendly facilities for loading drivers, opening and closing connections, retrieving database items, etc.?
5. Are there possibilities for retrieving data items in their original types rather than converting back from strings?
6. Are there facilities for navigating the fetched datasets?

4.2.5 Criteria for evaluating the suitability for teaching the file processing capabilities

In applications that do not require the special features of databases, the need for remote file handling capabilities is desirable for performance reasons. Section 3.7 of Chapter 3 discussed the file processing techniques in Web application development. This section is aimed at establishing the criteria that will ensure that the desired platform is capable of such file handling capabilities. These criteria are enumerated in detail below:

- ☑ Availability of commands or functions for creating files remotely on the Web server's disks: this demonstrates how to create files relative to client-based information, such as using client's surname as filename.
- ☑ Facilities for storing records sequentially on the files: this demonstrates the communication between the client-based information and the created files.
- ☑ Facilities for uploading files: this demonstrates the important concept of transferring whole files from clients to the server.
- ☑ Availability of variables that contain information about uploaded files: this illustrates where to get information about the uploaded files.
- ☑ Facility for setting the maximum value for the size of the uploaded files: this illustrates ways of avoiding uploading of excessive files that could affect the performance of the Web application negatively.

- ☑ Function for testing whether a file has been successfully uploaded: this illustrates how to ensure reliability of the uploading operation.
- ☑ Function for moving uploaded files from the server's default directory to the desired directory: this shows one of the ways of managing the uploaded files.

Emerging criteria questions: File processing capabilities

1. Are there commands or functions for creating, opening and closing files on the server's disks?
2. Are there facilities for checking the existence of the files on disk?
3. Are there facilities for writing or storing records sequentially on the files?
4. Are there facilities for reading records sequentially from the files?
5. Is there a file lock facility to prevent more than one script from accessing a file at the same time?
6. Are there facilities for uploading files from users' disks?
7. Are there facilities to contain information about the uploaded files?
8. Can the maximum size for the uploaded files be set?
9. Are there facilities to check whether a file has been uploaded successfully?
10. Are there facilities for moving uploaded files from the server's default directory to desired directories?

4.2.6 Criteria for evaluating the suitability for teaching exception handling

Exception handling is very important in achieving a robust application. In a Web application where one cannot predict the situation of a remote resource, exception handling will ensure the system's continued operation, despite the error being encountered. Section 3.8 of Chapter 3 discussed the various exception handling techniques in Web application development that should be taught to second year students. The criteria below are very important in ensuring suitability of the platforms for teaching exception handling techniques to second year students:

- ☑ Ability to extract the status code from the HTTP response: this is necessary to extract the status of the error detected by the server.
- ☑ Ability to translate error code and display pages that will present the errors in a meaningful language: this will assist the user in understanding what went wrong and to take corrective measures.
- ☑ Ability to specify the errors that the program is allowed to leave unhandled: this is necessary to allow the system to continue operations in situations of minor or expected errors.

- ☑ Ability to disable or suppress error messages for a single expression: this is also necessary to allow the system to continue operations in situations of minor or expected errors.
- ☑ Ability to turn off error reporting entirely, possibly temporarily, to enable initial concentration on the logic of the program: students need to get the logic right before handling syntax errors.
- ☑ Ability to intercept and handle run-time errors in the programs such that the program can recover from the errors: this illustrates how to handle runtime exceptions.
- ☑ Ability to exit gracefully without abruptly stopping the program; this illustrates how to perform necessary "housekeeping chores" which ensure consistency of program states.
- ☑ Ability to log error on files on disk: this demonstrates the techniques of keeping permanent records of errors encountered in order to gain more understanding of them and corrective actions later.

Emerging criteria questions: Exception handling capabilities

1. Is it possible to set and extract the status code representing error status stored by the server?
2. Does the platform translate error code and display pages that will present the errors in a meaningful language?
3. Are there facilities for specifying the errors that the program is allowed to leave unhandled?
4. Are there facilities for disabling or suppressing the error messages for a single expression?
5. Are there facilities for disabling or turning off error reporting completely?
6. Are there facilities for intercepting and handling run-time errors in the programs such that the program can recover from the errors?
7. Are there facilities for exiting gracefully without stopping the program abruptly, thereby ensuring consistency of program states?
8. Are there facilities for logging or recording errors on files on disks?

4.2.7 Criteria for evaluating the suitability for teaching the XML support capabilities

The importance of the capabilities for handling XML documents have already been stressed in section 3.9 of chapter 3. These are necessary to ensure interoperability among Web applications. The criteria for ensuring suitability of the platform for teaching XML document handling involve the ability to generate as well as to parse XML documents,

which serve as intermediate states of the information flowing between Web applications. These criteria are given in detail below:

- ☑ Ability to deliver XML documents rather than raw HTML to the browsers: this illustrates how to generate XML documents.
- ☑ Ability to attach a style sheet to the XML data: this is necessary in order to transform the XML document into a browser compatible HTML format.
- ☑ Facility for embedding an XML data island directly in an HTML page by referencing an XML document from the HTML document: to demonstrate interactions between HTML and XML documents.
- ☑ Referencing an XML data island in an external file from an HTML using the *SRC* attribute that specifies the path to the required XML file: to further demonstrate interactions between HTML and XML documents.

Emerging criteria questions: XML support

1. Is there the possibility of delivering XML documents rather than raw HTML files to the browser?
2. Is there support for attaching style sheets to the XML data?
3. Are there facilities for embedding an XML data island directly in an HTML page by using the XML element?
4. Is it possible to reference an XML data island in an external file from HTML using the SRC attribute?
5. Are there facilities for binding an XML data island to an HTML table by setting the DATAFLD attribute for each column (“TD”) in the table?

4.3 Criteria for evaluating the ease of use of the platforms

This section presents the desired qualities that will serve as criteria for use in evaluating the ease of use of the dynamic Web platforms. Being easy to use is here relative to the second year students of Web applications development specialization in the National Diploma in Information Technology at tertiary institutions. We will examine the availability of various features that will make the various tasks of Web application development much easier.

4.3.1 Availability of smart Integrated Development Environment

This includes the availability of interpreters and compilers, as well as the availability of smart integrated editing systems. According to Sebesta (1996, p.18), both the costs of

training programmers and of writing programs can be significantly reduced in a good programming environment. A programming environment is the collection of tools used in the development (Sebesta, 1996, p.31).

It is also important whether the interface to the tool is graphical, window-based and similar to common interfaces. Bergin (1996) states the need for a platform independent graphics model and a platform independent window-button-mouse based GUI. The following criteria are considered necessary in ensuring that our desired platform has a user-friendly and smart IDE:

- ☑ Availability of interpreters rather than compilers: for second year students, interpreters will be useful for quickly executing what they have done so far, and for considering the effects of minor changes to the programs.
- ☑ Availability of an IDE that is both DOS and Windows-based; this way, the system can be used easily in both DOS-based and Windows-based systems.
- ☑ Availability of smart integrated editing systems that:
  - enhance maintenance by encouraging good readability of the code by way of automatically formatting and indenting code;
  - are equipped with tracing and debugging aids to assist students in debugging their code;
  - offer other useful and smart editing facilities such as identifying errors as soon as commands are entered, rather than waiting till compilation time;
  - are equipped with context-sensitive helps to assist students who are "stuck" in certain activities; and
  - have the ability to switch case sensitivity on and off so that it can be turned on when there is the need for many related identifiers, such that their minor differences can be represented by alternating the case of the letters of the identifiers.

Emerging criteria questions: Smart IDEs

1. Does the platform use interpreters rather than compilers?
2. Is the IDE both DOS and Window based?
3. Does the IDE automatically create directories to store the project?

4. Does the IDE show the directory tree for the project?
5. Does the IDE allow menu-driven addition of Web items such as HTML pages, scripts, style sheets, etc.?
6. Does the IDE enhance maintenance by encouraging good readability by formatting and indenting code automatically?
7. Does the IDE have hotkey matching of braces, parenthesis, and angle brackets?
8. Does the IDE have customizable code formatting templates?
9. Is the IDE equipped with tracing and debugging aids to assist students in debugging their code?
10. Does the IDE offer other useful and smart editing facilities such as identifying errors as soon as commands are entered, rather than waiting till compilation time?
11. Does the IDE give syntax highlighting in configurable colours?
12. Does the IDE have the ability to switch case sensitivity on and off when necessary?
13. Does the IDE have function and variable name autocompletion?
14. Does the IDE have pop-up parameter references for recognized functions?
15. Does the IDE have an HTML toolbox for adding HTML components as well as WYSIWIG editing?
16. Does the IDE enhance database integration by having database tools such as viewing tables, adding connection objects?
17. Is the IDE equipped with context sensitive helps to assist students who are "stuck" in certain programming activities?
18. Does the IDE allow searching documentation and API from the help menu?
19. Does the IDE show split view of long scripts?
20. Does the IDE show different views such as the explorer class view, program view, debugging view and output view ?
21. Does the IDE have configurable keyboard shortcuts?
22. Does the IDE automatically construct the URL for the output?
23. Is the platform equipped with an independent GUI development tool?
24. Is there a class wizard to assist in the creating of classes?
25. Is it possible to start and stop the server from within the IDE?
26. Does the IDE support unlimited undo and redo?

4.3.2 Availability of features to simplify execution of programs

There are many activities involved in the execution of developed programs, and each activity may also involve many steps. These could take valuable time, which is often not available during a timed laboratory session. This section lists the criteria that will ensure that the series of activities to be carried out in the execution of programs are completed within reasonable time, and with the minimum of hardware resources.

Moreover, certain characteristics such as the Web server's software architecture, can affect performance significantly (Menasce, 2003). These criteria imply that the selected dynamic Web platforms will be suitable for the repeated compilation, testing and debugging in a student laboratory, and also that they are suitable for the size of computer affordable by students. These criteria as given below necessitate that the suitable platform will demonstrate the following attributes:

- ☑ Offer integration with other relevant components of the curriculum, such as integration to other Microsoft products, gaining momentum from such knowledge thereby speeding up the execution processes.
- ☑ Use minimum disk space and memory resources, such that files can be moved around easily, and execution can be faster.
- ☑ Have simpler installation steps. Fewer settings to be done on different configuration files will decrease the complications of executing the programs.
- ☑ Have simpler deployment requirements. Programs stored in fewer levels of directory relative to the 'c:\' root directory will facilitate speedy operations.
- ☑ Have simpler execution steps. Must programs first be compiled? Instant interpretation enables students to see results of code promptly.
- ☑ Have commands with syntax, semantics and meanings that will enhance comprehension, remembrance and correction of the commands.
- ☑ Have a reasonable amount of unique basic syntactical components, such that users can know most of the commands with ease.
- ☑ Allow for a high percentage of tasks to be completed per unit time by having simpler structures for programs.
- ☑ Reduce the time spent on errors, by pointing more directly to the error and giving useful diagnostic aids, rather than a terrifying display of error diagnostics.
- ☑ Demonstrate the availability of help and standard documentation, such that students have opportunities to check out correct syntax for statements
- ☑ Reduce the amount of time spent using help or documentation by offering examples of command usages.
- ☑ Have a number of good features offered by way of functions easily recalled by users, making it interesting for users.
- ☑ Have a number of useful features offered by way of functions easily recalled by users, making it interesting for users.
- ☑ Provide alternate ways to work around a problem, so as to avoid deadlocks during the course of program development.

- ☑ Demonstrate familiarity and consistency of commands and system identifiers, thereby enhancing their remembrance.
- ☑ Have generalized ways of forming command names thereby enhancing their remembrance.
- ☑ Offer flexible features to support important concepts such as multi-threading.
- ☑ Demonstrate robust support being provided to the user in determining successful achievement and assessment of goals. These include recoverability and opportunity for users to observe how the processing is progressing as well as responsiveness to confirm successful accomplishment of tasks.
- ☑ Have much higher-level commands or buttons to achieve some lengthy tasks, thereby reducing time spent in accomplishing specified tasks.
- ☑ Have simpler architecture to facilitate fast execution and speedy operations.
- ☑ Have a garbage collection facility to remove unused memory allocations, thereby availing memory space available for more data items per unit time leading to fast execution of programs.

Emerging criteria questions: Features to simplify execution

1. Does the platform integrate with other relevant components of the curriculum, such as the teaching of other programming languages?
2. Does the platform use relatively moderate disk space?
3. Does the platform have simple execution steps, i.e. are there fewer settings to be made before it works?
4. Does the platform have simpler deployment requirements, i.e. is the default directory in fewer levels relative to the "c:\" root directory?
5. Does the platform have a reasonable amount of unique basic syntactical components such that a user can learn most of the commands off by heart?
6. Does the platform reduce the time spent on errors, by pointing more directly to them?
7. Is there valuable help through standard documentations such that students can check out correct syntax of statements?
8. Does the documentation have a good number of example programs pre-installed?
9. Does the platform demonstrate "generalize-ability" and consistency of commands such that features and functions are easily recalled by users?
10. Is the platform flexible by offering multi-threading?
11. Are there many higher-level commands or buttons to achieve some lengthy tasks?
12. Does the platform have simple architecture?
13. Does the platform have a garbage collection facility?
14. Is it possible to avoid the need for the program to be compiled first before execution?
15. Are the processes of compilation, linking and execution done behind the scenes by the compiler / interpreter?
16. Can programs be executed both from the command line and from the browser?
17. Are array bounds not checked both at compile time and at runtime?
18. Does the platform have high performance in database retrieval applications?

Satisfaction of the above criteria will ensure that the chosen platform is fast to use. However, apart from these features, we consider it necessary to test the speed of executing the programs. This is so because while the program is executing, there is usually nothing more the student / programmer / tester can do than wait. A suitable platform should reduce this waiting period. The next section is devoted to the design of the latency experiment.

4.4 The latency experiment for evaluating the speed of execution

As mentioned by Leedy and Omrod (2001, p101), using two or more approaches enables us to learn more about the world than when we are limited to only one approach. Therefore this section supplements the criteria to ensure fast execution given in section 4.3.2 with the design of a latency experiment, allowing us to consider the problem from different angles. In other words, this study consists of an interaction between theoretical foundations and empirical evidence.

The latency experiment aims to estimate the execution speed for programs written using each of the dynamic Web platforms in order to gain information on the suitability of these platforms for recurrent program testing in a practical laboratory session. We will measure the performance together with scaling by estimating the response time while increasing the size of a database.

It should be pointed out that users of the framework can conduct their own latency experiment. Dehinbo (2005) presents the full details of this experiment, but below is a brief discussion of the latency experiment. We start by discussing the system configuration and the database configuration for the experiment. This is followed by the procedure for obtaining the latency data. It ends with the limitations of the experiment.

4.4.1 The system configuration for the latency experiment

The architecture adopted is a three-tier system, with the client browser connecting to the server which also hosts the Database Management System, as depicted in Figure 4.2 adapted from Dehinbo (2004a). The minimum system configuration needed to achieve this configuration is any computer system capable of being connected to the Internet and used as a Web server. The server used is a Pentium IV 2.4M.Hz computer with 256MB RAM connected to the Internet via LAN card to the University Network. For the platforms to be compared in the implementation, the appropriate software are installed.

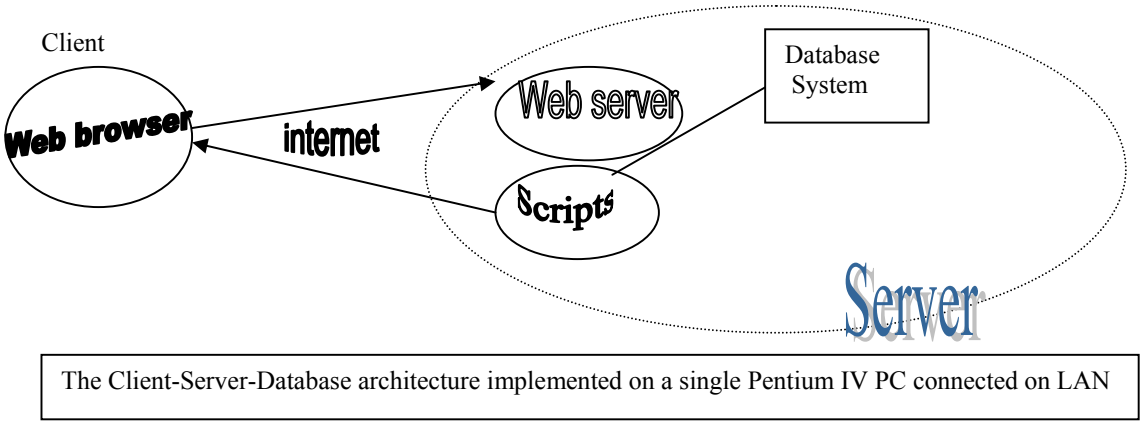

Figure 4-2. Typical configuration for a three-tier architecture

4.4.2 The database configuration

In order to estimate the time taken to access and display data from the database, a database will be created for all the platforms using Microsoft Access 2000. The database consists of the following structure:

Studnumber (int), surname (text), initials (text), sex (text), diploma (text)
subject1 (int), subject2 (int), subject3 (int), subject4 (int), subject5 (int), subject6 (int).

We will begin the program execution with 10000 records, and then double the size of the database to 20000, then to 40000 and then to 80000 records. This is simply because it is easier to duplicate the whole records than to enter new records.

4.4.3 The procedure to be followed to obtain the latency data

An algorithm is designed to access and display all the stored records in the database. This algorithm will be translated into a program to be written in the various platforms that will be compared in the testing of the framework. Each of these programs will extract the system time at the entry point of the program as “starttime”, and extract as “stoptime” the system time at the end of the program. The difference between the “starttime” and the “stoptime” in seconds is taken as the estimated time for the database retrieval operation. This is used as an estimate of the latency value for the dynamic Web platform. The experiment will be performed twice in order to use the average value.

In chapter 6 we will present the results of the implementation and rank the latency of the platforms obtained from the experiment. This ranking will then be assigned a weight and added to the scores for the implementation based on the criteria in sections 4.2, 4.3 and 4.5.

4.4.4 Limitations of the latency experiment

This experiment is limited to the use of the algorithm for database retrieval operations. This is because most Web applications access database (Adida, 1997), and the provision of the up-to-date information from database makes the application dynamic.

From experience, the author has noted that some platforms such as ASP lack functions to estimate time up to the microsecond level. The now ( ) function only obtains system time to the latest second. Therefore, we decided to increase the records until the time taken was at least 1 sec. We therefore started with 10,000 records for all the platforms.

4.5 Criteria for evaluating the cost of the platforms

Answers to the questions of determining suitable platforms must include analysis and understanding beyond technical merits, as there are several non-technical issues to consider, foremost of which is the cost effects (Vinoski, 2004). We agree that an influential factor in the comparisons of programming languages and Web platforms is the

cost of the platform. If a particular platform is prohibitively costly, there will not be any interest in choosing it for future works or for moving any previous works to such a platform, even if it is technically feasible.

The author agrees with Sebesta (1996, p.17) that the ultimate total cost of a programming language is a function of many of its characteristics as enumerated below. First is the cost of acquiring the platform, its Web server and related software such as the database management systems. Second is the cost of training programmers to use the language. This is a function of the simplicity of the language. Third is the cost of executing programs written in a language which is greatly influenced by that language's design. A language that requires many runtime checks will prohibit fast code execution.

The final consideration is the cost of maintaining programs, which includes both corrections and modifications to add new capabilities. Because the maintenance of software is often done by people different from its original authors, poor readability can make the task extremely challenging.

With the proliferation of various open source software, the cost of acquiring the platform, its Web server and related software will have an effect on the wide usage of the platform. This is even more important considering the fact that students have less money available to purchase software, yet they need to practice in their leisure time (McDougal & Boyle, 2004). Given that some of the cost factors have been addressed in other sections, the list below summarizes the criteria for evaluating the cost of acquiring the middleware, its Web server and related software:

- ☑ Is the dynamic Web platform available or downloadable free of charge?
- ☑ Is its associated Web server available or downloadable free of charge?
- ☑ Is the related software such as the DBMS available or downloadable free of charge?
- ☑ Are the Web server and the DBMS available together with the platform as a downloadable package, thereby enhancing their compatibility?

Emerging criteria questions: Cost

1. Is the platform available or downloadable free of charge?
2. Is the associated Web server available or downloadable free of charge?
3. Is the related software such as the DBMS and its driver programs available or downloadable free of charge?
4. Are the Web server and the DBMS available together with the platform as a downloadable package, thereby enhancing their compatibility?
5. Do the Web server and the DBMS, together with the platform, come free of charge together with purchased operating system or other software?

4.6 Criteria for evaluating the portability of the platforms

Portability is a factor that is very important when choosing a platform. This is even more crucial given the fact that these days, most programmers move from one organization to another and so knowledge and practical experience gained in one organization can be instantly made use of in another organization. Also, portability is needed for effective teamwork.

However, portability can be influenced by several factors. These include the architectural neutrality, applicability to a wide range of applications and the completeness and precision of the language's official defining document (Sebesta, 1996, p.19). Other factors include the programming environment comprising the set of operating systems, Database Management Systems and the Web servers which the dynamic Web platform can work with.

This section presents the qualities that will serve as criteria when evaluating the portability of the dynamic Web platforms. This will ensure that portability is given appropriate weight in the choice of the most suitable platform.

4.6.1 The criteria for evaluating the architectural neutrality, portability and familiarity

Architectural neutrality aims at freeing languages from implementation details (Wigglesworth, 2000, p.8) while portability is the ease with which programs can be moved from one implementation to another (Sebesta, 1996, p.19). According to Wigglesworth (2000, p.9), portability is the major benefit of architectural neutrality.

Portability is most strongly influenced by the degree of standardization of the language (Sebesta, 1996, p.19). The list below summarizes the criteria for evaluating the architectural neutrality and the portability of the dynamic Web platform:

- ☑ Is the platform architecture-neutral by allowing implementation-defined features? For example, do the target processor and the compiler determine the storage allocated for primitive types in the platform?
- ☑ Is the storage mapping for primitive types defined for all implementations on all target processors, thereby increasing portability?
- ☑ Are data types in the platform independent of the underlying hardware and operating system? For example, is an *int* data type always 32 bits, regardless of whether the code is executing on a 32-bit or a 64-bit CPU?
- ☑ Is the intermediate code portable by running on all implementations of the platform, thereby increasing portability?
- ☑ Has cross-platform development been a major goal in the development of the platform, thereby increasing portability?
- ☑ Is the main reason for running the platform on Windows to develop locally before deploying in a production environment such as Unix-based environments? This will ensure that knowledge gained by students using Windows will still be relevant in industrial situations.
- ☑ Can programs on the platform be used both on Intel-based Microcomputers as well as on Macintosh computers using the Mac Operating System though requiring some installation variations and some modifications to the code in order to port it to other implementations? This will also increase portability.
- ☑ Can the developed programs work across major operating systems without much modification to program code? This will definitely imply widespread usage of the platform, making the students more relevant and valuable in the industry.

The applicability to a wide range of applications and the completeness and precision of the language's official defining document are two other important criteria:

- ☑ Is the platform applicable to a wide range of applications from serving static Web pages, accessing Web databases in a two-tier application to acting as controller in three-tier applications?
- ☑ Are there various complete and precise official defining documents available freely on the Web?

Another factor that determines the degree of portability is the familiarity of the operating environment. It is very important that the interface to the tool is graphical, window-based and similar to common interfaces. Bergin (1996) highlighted the need for a platform independent graphics model and a platform independent window-button-mouse based Graphical User Interface (GUI). Another important issue on programming environment is the set of operating systems, Database Management Systems and Web servers which the dynamic Web platform can work with. The below questions could serve as criteria for evaluating the degree of familiarity of the operating environment for the platforms:

- ☑ Is the tool graphical, window-based and similar to common interfaces?
- ☑ Is the platform equipped with an independent graphics model and a platform independent window-button-mouse based GUI?
- ☑ Is the platform applicable to a set of different operating systems?
- ☑ Is the platform applicable to a set of different database management systems?
- ☑ Are there different Web servers which the Web-based middleware platform can work with?

Emerging criteria questions: Architectural neutrality and portability

1. Is the platform architecture-neutral by allowing implementation defined features e.g. do the target processor, operating system and the compiler determine the storage allocated for primitive data types?
2. Is the code portable by running on all implementations of the platform thereby increasing portability?
3. Can the code execute on platforms installed in both Intel-based and Macintosh also?
4. Is the platform restricted to using a specified Web server?
5. Is the platform restricted to using a specified DBMS?
6. Is the platform proprietary to a specific operating system?
7. Is the platform backward compatible?
8. Does the platform support interconnection and have facilities for talking to other services?

Again, positive answers to these criteria will enhance the platform's compatibility and portability. These allow more freedom to the users of such systems when choosing

accompanying technologies. Sections 4.2 to section 4.6 dealt with the various components of our framework. The next section puts all these together in the framework.

4.7 The framework and implementation of a tool to apply it

The previous sections dealt with the various components of the developed framework. This section puts the components together and describes the implementation of a spreadsheet tool to apply the framework.

4.7.1 Description of the framework

The developed framework consists of a series of criteria for evaluating the platforms, as well as a programming experiment for ranking the platforms in order of their performance. The various categories and subdivision of the criteria and experiment are provided in table 4.1 below.

The first set of criteria aims to measure the suitability for teaching the Web application development concepts. This is subdivided into criteria to measure suitability for teaching basic programming concepts and those for teaching other Web application development concepts. The basic concepts include lexical structures, data types, expressions and operators, control structures, and functional modularizations, while the other programming and Web development concepts include object-oriented programming, Web techniques (such as session management), database handling capabilities, file processing capabilities, exception handling capabilities and XML support.

The second set of criteria aims to measure the ease of use and the speed of use of the platforms. This is subdivided into criteria to measure the availability of desirable features of smart IDEs and features to simplify execution and decrease latency. To supplement the features that decrease latency, the latency experiment is included to rank the platforms in order of their latency. It is important to note, however, that users of the framework can conduct their own latency experiment, as long as it ranks the platforms. The platform

with the lowest latency is given a high score while that with the highest latency is given a low score. Users can then multiply these scores with weights as they find necessary.

The third set of criteria measures the cost and portability of the platforms. The cost criteria aim to ensure that a chosen platform is affordable by students such that they can install it easily on their PC at home and practice extensively. Portability criteria aim to ensure compatibility across major hardware specifications and operating systems. This increases the likelihood that work started in the laboratory can be continued on their home PC and can even be continued on their future office computer. The survey is simply included for informal documentation of past students' experiences on various Web platforms.

Table 4-1. Summary of the developed framework.

Criteria to measure the overall suitability for the teaching of Web development concepts: **Basic concepts** Lexical Structures Data types Expressions and Operators Control Structures Functions and modularizations **Other Web development concepts** Object-oriented programming Web techniques Database handling capabilities File processing capabilities Exception handling capabilities XML support
Criteria to measure the ease of use and speed of use Availability of smart Integrated Development Environment (IDE) Features to simplify execution and increase latency
Latency experiment to measure the speed of execution of program
Criteria to measure the affordability of the platforms and related components Evaluating the cost of acquiring the platform and related software
Criteria to measure the portability of the platforms Evaluating the architectural neutrality, portability, generality and the familiarity of the operating environment of the platform
Survey (Included in the Appendix to informally document the experience of past students taught using the platforms and see how it relates to the results of testing the framework)

4.7.2 Development of a tool to apply the framework in Microsoft Excel

Users must be able to manipulate the framework according to their preference. As an example, users should be able to vary the importance of some major categories of criteria relative to others. Also, should a user decide not to include a criteria, it should be easy to exclude such, observe its impact, and if necessary, re-include it and also vary its weight between 0 and 1. In order to allow for easy manipulation of the framework as desired, a tool is developed and implemented in the form of an Excel spreadsheet.

We use weights as a means of varying the importance or relevance of each major category of criteria and for each criterion also. A weight is attached to each of the criteria as well as the programming experiment, so that the user has the choice of excluding any of them by using a weight of zero or to increase their relative importance by increasing each weight relative to others.

At the lowest subdivision of the criteria, there is a series of questions that are used to evaluate the platforms. Each question has a relative weight such that users can use a weight of 0 or 1 to exclude or include the question respectively and vary its weight between 0 and 1. This implies that the total number of questions used within each sub division cannot be predicted. Therefore, a normalized weight is further calculated for each question as a proportion of that question to the total number of questions included.

The normalized weight calculated for each question is then used to multiply the response (1 to 3) for each question to obtain weighted responses for each of the platforms. The sum of these weighted responses is obtained for each platform under each lowest subdivision of the criteria.

Given that the number of questions varies for each lowest subdivision of the criteria, we find it necessary to ensure that each lowest subdivision is not given undue advantage because it has more questions than others. Therefore, we derive an average number of questions for each of the two major categories of criteria. This is obtained as the total

number of questions in all the subdivisions in each category divided by the total number of subdivisions in that category. This is then used to derive a normalization factor for each of the subdivisions of criteria in each of the major categories.

The normalization factor is calculated as the total number of questions for each category divided by the number of questions used in each lowest criteria subdivision. This yields a multiplying factor value of "1" when the number of questions in a subdivision is equal to the average number of questions in all the subdivisions. However, when the number of questions in a subdivision is high, it yields a value lower than 1 to reduce the effect of the high number of questions. Conversely, when the number of question in a subdivision is less, it yields a value higher than 1 to multiply the total response and increase the value thereby compensating for the low number of questions.

To obtain a normalized weighted total score for each platform under each subdivision of categories, the total weighted score or responses for that subdivision is multiplied by the normalization factor for each subdivision of categories. This is done for the major categories of criteria. These intermediate results for each platform under each subdivision in the major categories are fed into another spreadsheet to obtain summary data for the framework. The whole processes so far is illustrated in the simulated spreadsheet in the table 4.2 below in which the formulas, rows, columns, cells (intersection of row & column) are given using Microsoft Excel format.

Table 4-2. Spreadsheet implementation for each criterion

Column	A	B	C	D	E	F	G	H
Row number	Category's subdivision	Question	Question's weight set by user	Normalized weight	Response for platform X	Weighted response for Platform X	Response for platform Y	Weighted response for Platform Y
1	Data types	Q1=….	1	=C1/C4	3	=D1*E1	1	=D1*G1
2	(as an	Q2=….	1	=C2/C4	1	=D2*E2	2	=D1*G1
3	example)	Q3=….	0	=C3/C4	2	=D3*E3	3	=D1*G1
4	Total		C1+C2+C3			F1+F2+F3		H1+H2+H3
5	Weighted score Total					=F4 x Nor-malization factor		=H4 x Nor-malization factor
…	…..	……	……..	……	……	……..	……	……

Before obtaining summary data for the framework, we consider it necessary to allow users to vary the relative weight of each of the lowest subdivision of categories. This enables one to give more importance to desired concepts. Let us consider an example within the criteria for measuring suitability for teaching Web application development concepts under the subsection for basic concepts. One might consider reducing the weight for the lowest subsections such as Lexical structures, data types, and Functional modularization while increasing that of expressions and operators and control structures.

In order to free the user from the responsibility of ensuring the proportionality of the allocated weights as a percentage, we allow users simply to enter weights for each lowest subsection relative to that of others. Therefore, a normalized weight is further calculated for each lowest subsection as a proportion of the weight for that subsection to the total sum of weights of all lowest subdivisions in that subsection. This normalized weight is then used to multiply the intermediate results fed in from the spreadsheet containing values for each platform under each subdivision in the major categories.

Finally, we still consider it necessary to allow users to the possibility of varying the relative weight of each of the major subdivisions of the categories. This enables one to give relative importance to the suitability for teaching basic concepts, other concepts, as well as the ease and speed of use, and also to the latency experiment. This ensures, for example, that anyone not interested in including the latency experiment can simply assign a weight of zero (0) to it. Again, we derive and use proportional weights based on the ratio of each weight to the total sum of weights. Final values are then obtained for each platform after ensuring the effects due to the proportional weights. An illustration in a simulated Excel worksheet is given in table 4.3 below.

Eventually, for all the major subdivisions of the categories (suitability for teaching basic concepts, other concepts, as well as the ease and speed of use, and also the latency experiment) whose weight is greater than zero, the previous final values obtained for each platform are summed up. The platform that emerges with the highest value will be the

one that the framework determines to be most suitable for teaching Web application development in tertiary institutions.

Table 4-3. Spreadsheet implementation for the general summary data computation

Column	A	B	C	D	E	F	G	H
Row number	Main criteria's Category	Category weight set by user	Category's Subdivision	Category's subdivision weight	Normalized subdivision's weight	Previously Calculated value for Platform X	Weighted value for Platform X	Previously Calculated value for Platform Y
1	Basic concept	25	Lexical structure	20	=D1/D6	=Sheet1!F5	=E1*F1	=Sheet1!H
2	(as an		Data types	15	=D2/D6	=Sheet1!F	=E2*F2	=Sheet1!H
3	example)		Expressions & operators	20	=D3/D6	=Sheet1!F	=E3*F3	=Sheet1!H
4			Control structures	25	=D4/D6	=Sheet1!F	=E4*F4	=Sheet1!H
5			Functions & modlzatn	20	=D5/D6	=Sheet1!F	=E5*F5	=Sheet1!H
6	Total			Sum(D1:D6)			Sum(G1:G6)	
7	Weighted score Total						=G6 x B1 (Sum x category wt)	
…	…..	……	……..	……	……	……..	……	……

4.8 Summary, conclusions and link to the next chapter

This chapter describes the components of the framework for evaluating the dynamic Web platforms. The first section contains the criteria for evaluating the suitability of the platforms for teaching the identified concepts of Web application development. This is to establish that the platform is actually capable of satisfying the purpose of teaching Web application development concepts to second year students.

An important assumption made here is that, for second year students, factors that will enhance readability of the code will enhance comprehension of the code. Our priority here is to get the students started enthusiastically, and as they progress in their studies, other standards of software developments can be phased in. Otherwise students could

become disillusioned with dynamic Web applications development, create a low self esteem and achieve low pass rates.

The second section looks at criteria that will ensure that the Web application development tasks are carried out using applicable IDE that is user-friendly and easy to use. Closely related to this is the availability of features and simple architecture to facilitate fast execution and speedy operations. This is supplemented by an experiment that results in the ranking of the latency or speed of operations of the platforms. Fast execution and speedy operations will ease the recurrent compiling-testing-debugging activities in students' laboratory sessions.

The third section considers various criteria that will ensure that work being done by the students in the classroom can be continued at home, and can even be continued in the workplace after graduation. Central to these requirements is the cost and portability of the platform. A freely downloadable platform will definitely be within the reach of many students. Minimum hardware requirements for such platforms, as well as portability across major operating systems, will make the platform available to a greater number of students. The last section presents a summary of the framework. In addition, we describe the development of a tool to apply the framework in Microsoft Excel.

In summary, a selected platform resulting from the use of the developed framework will support both structured and object-oriented programming capabilities, with a user-friendly IDE. It should also provide some support for Web development techniques, exception handling, database manipulations and file handling, as well as XML parsing. In addition, the platform should be affordable, less resource-intensive, fast, and also portable to enable the continuity of tasks started in the classrooms, thereby increasing students' knowledge of Web application development.

With the framework in place, the next two chapters present the results of testing our framework using evaluations to arrive at our final choice. Ultimately, all the results are combined in the concluding chapter.

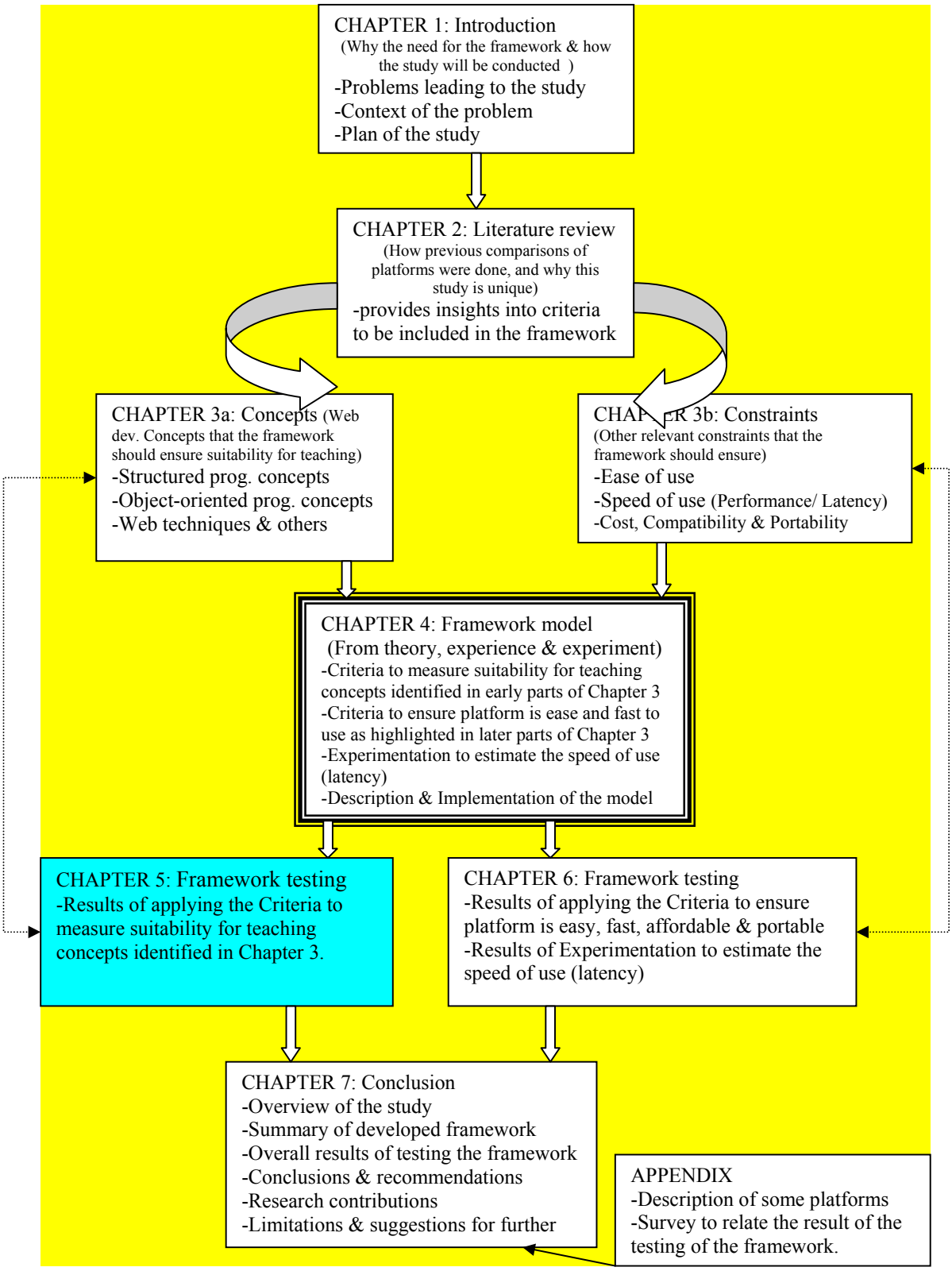

Figure 5-0. Dissertation map highlighting the relativity of chapter 5 to the rest of the dissertation.

CHAPTER 5

5 RESULTS FOR THE SUITABILITY OF THE WEB PLATFORMS USING THE CRITERIA BASED ON THE CONCEPTS OF WEB APPLICATION DEVELOPMENT

5.1 Introduction

In chapter 4, we presented the framework consisting of the various criteria and experiment, as well as the constraints that would be used to evaluate the suitability of dynamic Web platforms for teaching Web application development. This chapter presents the results of testing the framework by evaluating four selected platforms, with respect to the criteria set in section 4.2.

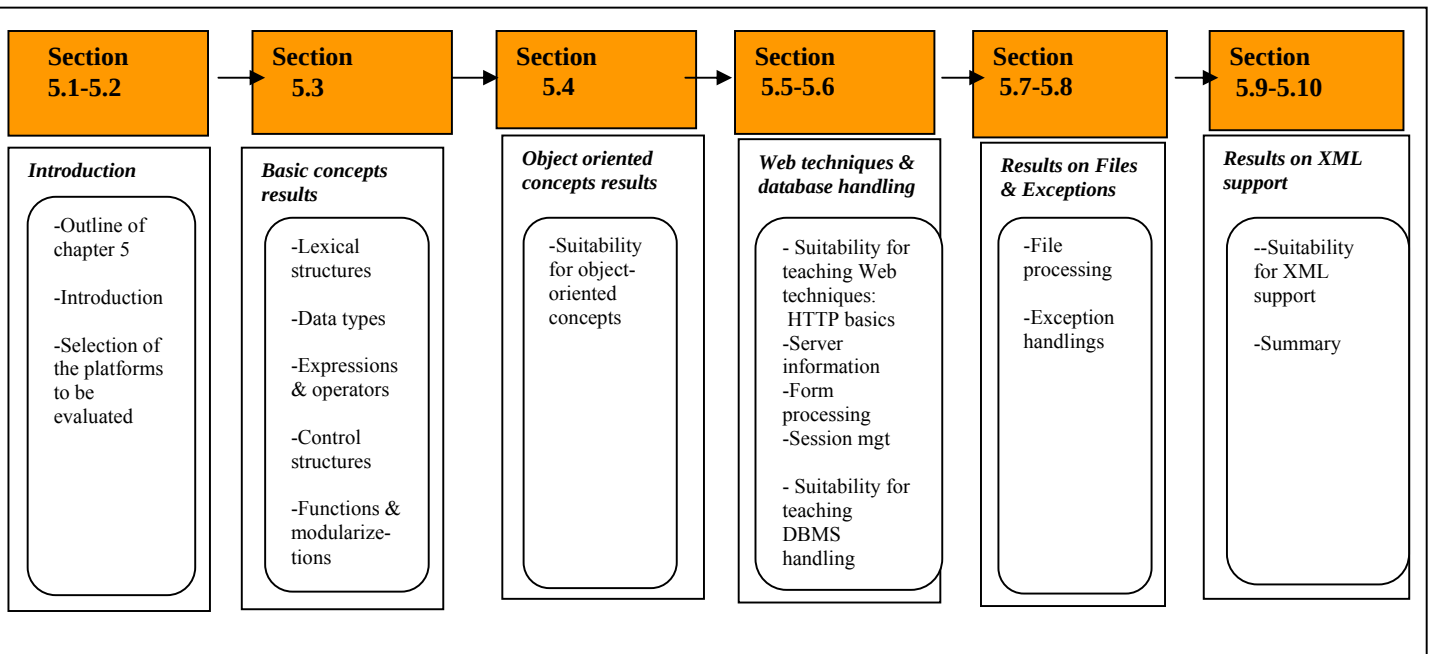

Figure 5.1. Pictorial outline of chapter 5.

As illustrated in figure 5.1 above, section 5.2 summarizes the selection of platforms to be evaluated. Section 5.3 includes the results on the criteria based on the basic concepts

(lexical structure, data types, control structure, functions). Section 5.4 to section 5.9 present the results on the other concepts (object oriented programming, Web techniques, file processing, database handling, exception handling and XML support) that are important for teaching Web application development in tertiary institutions. Lastly, section 5.10 presents a summary of the chapter and a link to the next chapter.

When asking the questions serving as the criteria of the framework later, we use a close-ended "Yes/No" questions, on a scale of **1 to 3;**

where 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

To answer the questions when evaluating the platforms, we used referenced information from textbooks, journals, authoritative web sites, as well as using author's observation and experiences. However, users of the framework can use their own references and experiences, especially if evaluating platforms different from the four we select due to the reasons below.

5.2 The platforms to be evaluated

With the framework in place, the next immediate step is the selection of the platforms that are to be evaluated. While users can test the framework using the platforms of their choice, we find it necessary to provide in Appendix 1 a brief description of some platforms that are commonly used for teaching and in the industry. We then provide reasons for not using some of the platforms in testing the framework, thereby narrowing down our choice of platforms.

One of the important constraints in the determination of a suitable platform for teaching dynamic Web applications is that it should be inexpensive to be available or affordable for personal use by students. This constraint excludes ASP.NET from the platforms to be evaluated in this study. The .NET framework containing ASP.NET is not available as part of the Windows XP/2000 environment unlike ASP. It is to be purchased separately and is expensive (Ervin, 2000; Microsoft, 2002, p1). It is also memory and disk intensive, meaning that it will not work properly on the cheaper computers which are all most students can afford.

PERL is used with the Common Gateway Interface (CGI), and is thus associated with the limitations of the CGI that every interaction creates a new process, leading to a slow and an inefficient implementation (Deitel *et al*., 2001, p.963; Westman, 2002). Therefore, PERL will not be used in this study.

Python will not be used in this study either as there is limited peer support for Python programmers on the Windows platform. This is because, as stated by Python Software Foundation (2004) as well as Rossum and Drake (2003), most Python developers are on Unix though there are a number of Windows/MacOS programmers.

Tcl will not be used in this study because of its need to be embedded in other applications in order to harness its full potentials. Even though the embeddability feature is advantageous (Apache, 2004; Waclena, 1997), it will not be suitable for the second year students due to the complexity of harmonizing its interaction with other applications.

ColdFussion will not be used in this study because it is only available free on a 30-day trial, after which one has to purchase it (Puck, 2004). Most students will not be able to afford that. Moreover, according to Westman (2002, p.7), ColdFussion uses the ColdFussion Markup Languages (CMFL) tags which look like HTML tags. This may make it difficult for students to separate logic from the presentation in HTML.

The platforms that will be used in this study are those that are popular with the Windows environment, inexpensive or available free either on the Web or on CD, and include a reasonable amount of documentation in the form of books or online materials. These features (Clark, 1999; jGuru, 2000; Karpinski, 1998; Ross et al., 2000; Sun Educational services, 2000; Sun Microsystems, 1997; Yager, 2001) are necessary for students' own private practice at home. This will enhance adequate comprehension and practice of the concepts. With this in mind, the selected platforms are Java Servlets, Java Server Pages (JSP), Active Server Pages (ASP) and Personal Home Page (PHP) Hypertext Processor. The next section presents the evaluation of the four selected platforms.

5.3 Evaluation of the suitability for teaching basic concepts

The results for the testing of the framework when using criteria set to evaluate the suitability of the platforms for the teaching of basic concepts of Web application development are sub-divided into the results on lexical structure, data types, expressions and operators, control structures as well as functions and modularization.

5.3.1 Lexical structure

Table 5.1 below shows the scoring for the platforms based on the criteria on lexical structures. This is followed its summary developed from the review of relevant literature and from programming experience.

According to Wigglesworth (2000, p.66), the convention in Java of naming variable names is to use lowercase for letters, but when the name is more than one word, the first letter in all words forming the compound word, except the first word, is capitalized for readability. An example is "timeOfDay" where 'O' and 'D' are capitalized but 't' is not. In ASP, there is no such general convention because there is no case sensitivity.

In PHP, we observe that variable names are usually in lower case but must be preceded by a dollar sign, which helps to identify variable names easily. Interestingly, as a mode of identification, Wigglesworth (2000, p.66) states that Java conversely recommends avoiding starting identifiers with a dollar sign as that usage is reserved for code that is generated automatically rather than manually. The use of the underscore " _" in names in Java, ASP and PHP also assists in forming expressive names (Bakken *et al.*, 2002b, p.33; Deitel *et al.*, 2001, p.728).

Names in the Java Servlet and JSP platforms are case sensitive (Wigglesworth, 2000, p.66). This also applies to commands and keywords. Conversely, in ASP using VBScript both names and commands are case insensitive (Deitel *et al.*, 2001, p.728). For PHP,

Lerdorf and Tatroe (2002, p.17) state that the names of user-defined classes and functions, as well as built-in constructs and keywords such as echo, while, class *etc*. are case-insensitive. However, variable names are case-sensitive in PHP. The case insensitivity in ASP and PHP will make coding easier for students.

Table 5-1. Scoring for the platforms based on the criteria on lexical structures.

		Servlet	JSP	ASP	PHP
1	Are there adopted ways for forming variable names from compound words, such that names can easily be remembered? e.g. timeOfDay ?	3	3	2	2
2	Are the formed variable names highly case in-sensitive? e.g. timeOfDay not different from timeOfday?	1	1	3	1
3	Are the keywords and commands highly case in-sensitive, e.g. print not different from Print?	1	1	3	3
4	Is there no compulsory punctuation used to end the statements, e.g. no need for using ; to end statements?	1	1	3	1
5	Are multiple statements disallowed on a line even though separated by special characters? e.g. print a * b : calculate c / d?	3	3	3	3
6	Are there special words to mark different compound statements (e.g. if … endif) instead of just using { and }?	1	1	3	3
7	Does the appearance of some statements and commands partially indicate their purpose, meaning or syntax? e.g. getString()?	3	3	3	3
8	Can one spread a statement across any number of lines instead of being forced to have one very long line that is often out of sight ?	3	3	2	3
9	Is it required to indicate when the next line is a continuation of the previous line, or is there an easy way of knowing?	2	2	3	2
10	Can one go straight into coding without many detailed requirements to declare and define class and service method structures as well as output objects?	1	3	3	3
	TOTAL	19	21	28	24

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

Java uses a statement terminator such as the semicolon (;) to end statements (Deitel *et al.*, 2001, p.732) as in PHP (Lerdorf & Tatroe, 2002, p.18), whereas VBScript in ASP does not. The lack of a terminator symbol is desirable for second year students, as forgetting to put the semicolon can cause errors not easily identifiable. Worst still, some compilers will only point to the error as being in the next line after the line without the statement

terminator. A novice programmer could spend hours looking at this otherwise correct statement before realizing that the error is the missing terminator in the previous line.

All the platforms disallow having multiple statements on the same line even if they are separated by special characters. This is good for error isolation, as students can easily identify which statement has the syntax error.

Java uses C-type braces (**{** and **}**) to group statements together (Sebesta, 1996, p.13). This has the effect that statement groups are always terminated in the same way, which makes it difficult to determine which group is being ended when an '**}**' appears. There are two methods of forming compound statements or statement groups in PHP (Bakken *et al.*, 2002b, p.62). It uses C-type braces (**{** and **}**) but also offers an alternative syntax for some of its control structures. It thus makes things clearer by using a distinct closing syntax for each type of statement group, namely, *if*, *while*, *for*, *foreach*, and *switch*. In each case, the basic form of the alternate syntax is to change the opening brace to a colon (:) and the closing brace to *endif*, *endwhile;*, *endfor;*, *endforeach;*, and *endswitch;*, respectively (Bakken *et al.*, 2002b, p.62).

Like PHP, ASP also uses keywords (e.g. End If, Loop, etc.) to group compound statements (Deitel *et al.*, 2001, p.728). This feature in both PHP and ASP is most useful for students since it makes the program more readable.

Designing statements so that their appearance at least partially indicates their purpose is an obvious aid to readability. Semantics, or meaning, should follow directly from syntax, or form (Sebesta, 1996, p.9). All the platforms evaluated use statements that indicate their purpose, thereby making it easier to remember the commands.

All the platforms evaluated allow one to spread a statement across any number of lines such that alignments and indenting can be used to aid readability. However, only ASP requires one to explicitly indicate when the next line is a continuation of previous lines.

This is useful for readability purposes. However, other platforms seem to cover their deficiency in this regard by using statement terminators.

Bakken *et al.* (2002b, p.2) show that in PHP, one can go straight into coding within HTML by simply enclosing the statements within "<% and %>". This is also true for ASP and PHP. However, in Java Servlet, we observed that one first has to declare a class and service method structure and output objects through which the HTML are sent to the client browser. This is more cumbersome.

5.3.2 Data Types

Table 5.2 below gives the actual scoring for the platforms based on the criteria on data types. It is followed by a summary of the results on data types for the four platforms evaluated, formed from the review of various literature and from programming experience.

Table 5-2. Scoring for the platforms based on the criteria on data types.

		Servlet	JSP	ASP	PHP
1	Is there "variant" data type so that a variable can be assigned values of different types, e.g. a variable used for float or string later in same program?	1	1	3	3
2	Is there support for overflow, e.g. are values growing beyond the bounds for int type interpreted as float instead of yielding errors?	1	1	3	3
3	Is the range for data type values high enough to reduce the situations for overflow?	3	3	2	3
4	Is there no (or very high) limit for the size of String values that can be assigned to a String variable, such that there is no concern about long strings?	3	3	3	3
5	Is the size of an array flexible and extendable so that it depends on run time values assigned?	1	1	3	3
6	Is there flexibility in using either of the parenthesis "( )" or "[ ]" in array processing?	1	1	1	3
7	Are there iterator functions for traversing array elements?	1	1	1	3
8	Can one cast or mix data types in an expression, when necessary, e.g adding int and float?	1	1	3	3
9	Is there flexibility to optionally declare variables or not, before being use in the program?	1	1	3	3
	TOTAL	13	13	22	27

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

According to Wigglesworth (2000, p.106), the Java platform is a *strongly typed* language and there are thus distinctly different *data types*. For ASP, Deitel *et al.* (2001, p.730) indicate that VBScript has only one data type, which is the *variant* data type that is capable of storing different types of data (e.g., strings, integers, floating-point numbers etc.). VBScript interprets a *variant* in a manner that is suitable to the type of data it contains. For example, if a variant contains numeric information, it will be treated as a number; if it contains string information, it will be treated as a string (Deitel *et al.*, 2001, p.730). Because all variables are of the *variant* type, the programmer does not specify a *data type* when declaring a variable in VBScript. In fact, variables can be declared by simply using their name in the VBScript code as they can still assume new data types. This allows flexibility for students.

A side effect is that if a variable name is misspelled, it is considered as a new variable, usually resulting in errors, which are sometimes very subtle. It could also lead to less readability. Fortunately, however, in ASP, the statement **Option Explicit** can be used to force all variables to be declared before they are used (Deitel *et al.*, 2001, p.731).

Similar to ASP, according to Bakken *et al.* (2002b, p.5-33), PHP also uses the variant *data type*. It supports eight primitive types, but the type of a variable is usually not set by the programmer; rather, it is decided at runtime by PHP depending on the context in which that variable is used. Thus, a variable may be evaluated with different values in certain situations, depending on what type it is at that time.

The fact that Java is *strongly typed* implies that variable declaration associates an identifier name with a type. Such an identifier cannot accept new type (Van Hoff, 1997; Wigglesworth, 2000, p.64). Therefore, there can be no integer overflow. If one specifies a number beyond the bounds of the integer type, it will not be interpreted as a float. Conversely, the weak typing in ASP and PHP allows for overflow. Lerdorf and Tatroe (2002, p.24) state that attempts to store a “too-large” integer variable will lead to storage of a floating point number. This feature in ASP and PHP is useful for beginner students.

To address the overflow problem above, the data range in most of the platforms is very high, with acceptable integers being as high as 2,147,483,647 (Lerdorf & Tatroe, 2002, p.24; Wigglesworth, 2000, p.110), except for ASP whose integer type is limited to 32767. It however uses *long integer type* to store up to 2,147,483,647 (Deitel *et al*., 2001, p.731). This high limit for integers will reduce the situations of overflows in students' programs.

Again, the fact that Java is *strongly typed* implies that one cannot mix *data types* in an expression (Wigglesworth, 2000, p.64). A float cannot be added to an integer without loss of precision. This is possible in ASP and PHP due to their *variant data type*.

Similarly, the limit for the size of *Strings* is very high in all the platforms. No limit is specified for *Strings* in major textbooks for the Java-based platforms. In ASP, *Strings* can take up to 2,000,000,000 characters (Deitel *et al*., 2001, p.731). In the use of PHP, Bakken *et al.* (2002b, p.12) state that there is no practical bound to the size of *Strings*, so a very long *String* does not present a problem.

Checking *array-bounds* implies that the size of an array is not extendable in the Java-based platforms and this eliminates a source of programmer errors (Van Hoff, 1997). According to Wigglesworth (2000, p.121), after the array has been created, one cannot extend it in the Java platforms. In PHP, values can be inserted into the end of an existing array (Lerdorf & Tatroe, 2002, p.24). In ASP, dynamic arrays whose size can change during execution are available in addition to fixed-size arrays (Deitel *et al.*, 2001, p.745).

The use of arrays is very important in programming. Yet a common source of error is to use the wrong parenthesis for specifying the subscript. This is common particularly among those who have used another platform that uses a different parenthesis. To overcome this problem, PHP, unlike other platforms evaluated, allows the use of both the "( )" as well as the "[]" (Lerdorf & Tatroe, 2002, p.118).

Despite the importance of array processing, the author observes that a limitation in most programming languages is that one has to traverse the array elements sequentially. PHP, unlike other platforms evaluated, provides various functions for traversing or iterating the array elements. These functions, according to Lerdorf and Tatroe (2002, p.126) include *current(), reset(), next(), prev(), end(), each()* and *key().*

5.3.3 Expressions and Operators

Table 5.3 below gives the scoring for the platforms based on the criteria on expressions and operators. A summary of the results on expressions and operators for the four platforms evaluated and formed from the review of various literature and from programming experience follows below.

The Java platform supports the C-type combined operator-assignment expressions such as "*expression+=3"*. According to DevDaily (1998), there are four special assignments defined to abbreviate common increments and decrements. These are ++ (add one to variable), -- (subtract one from variable), += *exp* (add exp to variable), and -= *exp* (subtract *exp* from variable). This is applicable to other operators such as * and /. It is also applicable to the *String* concatenation operator ("+") giving ("+=") for *String* incremental concatenation. It also supports the C-style pre- and post-increment and decrement forms; " ++a", "a++", "- -a" and "a--" (Wigglesworth, 2000, p.69).

PHP supports the C-type combined operator-assignment expressions except that the *String* concatenation operator (".") gives (".=") (Bakken *et al*., 2002b, p.59). It also supports the C-style pre- and post-increment and decrement operators "++a", "a++", "—a" and "a- -" (Bakken *et al.*, 2002b, p.57).

As mentioned earlier, PHP uses the + operator as an array operator. It appends the right handed array to the left handed, whereas duplicated keys are NOT overwritten (Bakken *et al.*, 2002b, p.60). As mentioned earlier, in other languages this would involve the use of a

loop structure. Thus the array operator in PHP is relatively more convenient than cumbersome loop operations.

Table 5-3. Scoring for the platforms based on the criteria on expressions and operators.

		Servlet	JSP	ASP	PHP
1	Are there relatively simple and convenient, rather than cumbersome, ways of specifying computations, e.g. using the notation count++ instead of count = count + 1?	3	3	2	3
2	Are the symbols used for arithmetic, logic and relational operators meaningful such that it is easy to remember what the symbol represents?	3	3	3	3
3	Are the relational keywords highly case in-sensitive, e.g. and same as AND?	1	1	3	3
4	Is operator overloading discouraged such that the same symbol cannot be redefined for use in different arithmetic and logic operations?	3	3	3	3
5	Does the division operator always give the same result for Integer division and Floating point division?	1	1	3	3
6	Is there a separate operator that can be used when it only integer division is desired?	1	1	3	1
7	Is the use of same symbol for addition and string concatenation discouraged?	1	1	1	3
8	Is there any special high level operator for exponentiation?	1	1	3	1
9	Is the assignment operator different from the equality operator?	3	3	1	3
10	Is the operator for string comparison the same as the operator for numeric comparison?	1	1	3	3
	TOTAL	18	18	25	26

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

These powerful operators make programs more compact and readable and students should know about them. Unfortunately, ASP does not support the use of the auto increment operators as such are not found in major ASP textbooks.

The symbols used for arithmetic, logic and relational operators are meaningful in all the platforms, being mostly more or less the same symbols as those used in mathematics. However, ASP and PHP have the advantage of the case insensitivity of logic operators such that, for example, “AND” is the same as “and”. This makes room for easy programming by students.

The overloading of operators, where an operator is redefined and given a new meaning, could pose problems for beginning students. Sebesta (1996, p.9) states that although this is a useful feature, it can lead to reduced readability if users create their own overloading and do not use it sensibly. This opinion is supported by Sun Microsystems (2001) which notes that although operator overloading enhances flexibility, it can be abused to generate hard-to-read code by redefining familiar symbols into intuitive operations.

A good example of a situation where operator overloading is dangerous is when a "+" is overloaded to mean subtraction. Readers will have difficulty in picking this up. Fortunately, however, all the platforms evaluated discourage the redefinition of operators. Java neither allows operators overloading nor allows one to add new operators (Van Hoff, 1997). Likewise, in PHP, no operator overloading is available (Apache, 2004).

Similarly, inbuilt operator overloading, in which a single operator symbol has more than one meaning, could be problematic for students. In the Java platforms, the division operator can be an integer division or a floating point division depending on its literals. According to Farrell (2003, p.36), when two integers are divided, the result is an integer. In other words, any fractional part of the result is lost. For example, the result of 9 / 4 is 2, even though the result is 2.25 in a mathematical expression. To get real value as in a mathematical expression, one has to use floating values such as 9.0 / 4.0 to produce 2.25 as answer. Students could spend hours trying to figure out such problems in code.

There is no integer division operator in PHP. Dividing 1 by 2 (½) yields the float 0.5 although you can cast the value to an integer to round it downwards or you can use the "round()" function (Bakken *et al.*, 2002b, p.10). However, in ASP using VBScript, unlike in Java, the division "/" is not overloaded as the separate integer division operator "\" exists (Deitel *et al.*, 2001, p.728). Therefore, when strict integer division is desired the "\" operator can be used. This avoids confusing students.

Still on the subject of operator overloading, the use of the same symbol for addition and String concatenation should be discouraged. The Java-based platforms use "+" for both

addition and String concatenation (Wigglesworth, 2000, p.23). Similarly, according to Deitel *et al.* (2001, p.729), VBScript provides the plus sign, "+", and ampersand, "&", operators for *String* concatenation. Given the traditional use of "+" for addition, the above means that the "+" operator is overloaded to function for both numeric addition and *String* concatenation. PHP, however, outshines others in this regard by using a separate *String* concatenation operator (Lerdorf & Tatroe, 2002, p.38).

The same goes for the need to separate the assignment operator from the equality operator. The Java-based platforms as well as PHP use "=" for assignment while "==" is used for testing equality of numerical values (Lerdorf & Tatroe, 2002, p.35; Wigglesworth, 2000, p.69). However, ASP using VBScript uses "=" as both the assignment operator and the equality operator.

The converse, however, is the case with regard to separating the operator for *String* comparison from that for numeric comparison. It seems better to use the same symbol for both. This is the approach taken by both ASP and PHP (Deitel *et al.*, 2001, p.729; Lerdorf & Tatroe, 2002, p.35). However, the Java-based platforms uses "==" for numeric equality comparison and ".**equals**" method for string equality comparison, as the "==" indicates only whether the object references are equal (Wigglesworth, 2000, p.131). Being different from other programming languages in this regard, even experienced programmers sometimes fall into this trap in the Java-based platforms.

5.3.4 Control structures

Table 5.4 below reflects the scoring for the platforms based on the criteria on control structures. From the review of various literature and from programming experience, a summary of the results for the four platforms evaluated also follows.

According to Sebesta (1996, p.12), the structured programming revolution of the 1970s made it clear that the indiscriminate use of "goto" statements severely reduces program readability. A program that can be read from top to bottom is much easier to understand than one that requires the reader to jump from one statement to some nonadjacent

statement in order to follow the order of execution. Most programming languages designed since the late 1960s, however, have included sufficient control statements such that the need for the "goto" statement has been nearly eliminated (Sebesta, 1996, p.12). This is applicable to all the platforms evaluated here.

Table 5-4. Scoring for the platforms based on the criteria on control structures.

		Servlet	JSP	ASP	PHP
1	Are there sufficient control statements such that the need for the unconditional branching (such as "goto" statement) has been eliminated?	3	3	3	3
2	Is there a special looping control structure for traversing array elements?	1	1	1	3
3	Are there possibilities for using both integers and strings as array subscripts?	1	1	1	3
4	Is the increment clause in a "repetition/loop structure" specified in clear wording?	1	1	3	1
5	Is there a default value for incrementing or decrementing a loop control variable?	1	1	3	1
6	Is the changing of the value of the loop control variable within the loop disallowed?	1	1	3	1
7	Is the omission of the "break" statement in a multiple selection statement not likely to cause disastrous logical errors of one "case" running into others?	1	1	3	1
8	Is it possible to use any other type, such as "string", as the control variable for the multiple selection statement?	1	1	3	3
9	Is the control structure block surrounded with meaningful keywords such as if … endif?	1	1	3	3
	TOTAL	11	11	23	19

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

Looping is very important in traversing array contents. In addition to the "for" statement in other platforms and in most languages, PHP also provides the "foreach" statement to iterate over elements in an array (Lerdorf & Tatroe, 2002, p.53). This is most useful in looping over array to access both the key and value of the array.

While traversing over elements in an array, the variable controlling the loop is often used as a subscript index to reference the array element. This is usually an integer in most languages, including the Java-based platforms. Deitel *et al.* (2001, p.745) state that in ASP using VBScript, any floating-point number is rounded to the nearest whole number

if used as an index. However, in PHP, array can use both integer and string indices (Bakken *et al.*, 2002b, p.22). An advantage of the String index is that it is more descriptive, and this is useful for students.

Experience shows that the increment of the loop control variable is an important factor in effective utilization of a loop control structure. It is therefore important for beginning students that this increment is specified clearly. Unlike other platforms that use the "++" in the increment of a "for" loop, only ASP uses the optional "STEP" keyword or clause to indicate an increment or decrement (Deitel *et al.*, 2001, p.734). This is definitely more readable. Moreover, when the "STEP" clause is omitted, the increment or decrement defaults to units of 1 (Deitel *et al.*, 2001, p.734). This default setting has the additional effect of reducing program size, which is also something that students will find useful.

Program testing shows that some calculations within a loop may result in changing the value of the loop variable or the loop condition. While this may not be obvious to novice programmers, it will have the effect of terminating the loop unexpectedly. Unlike the case in other platforms evaluated, in ASP using VBScript the "For" repetition structure's condition cannot be changed during the loop's iteration (Deitel *et al.*, 2001, p.734).

It has been observed that the multiple selection statement is a very important control structure as it elegantly avoids the need for using nested "if" statements. In most languages and platforms, forgetting to insert the "break" statement could cause logic errors that are not easily detectable. However, in ASP using VBScript, the "Select Case … End Select" structure does not require the use of a statement like "break". One **Case** cannot accidentally run into another. Moreover, unlike in the Java-based platforms, any variant subtype can be used with the "Select Case … End Select" structure (Deitel *et al.*, 2001, p.733). This is also true for PHP as it also supports the variant data type.

Conditional control structures usually test the value of a condition before taking the required action. Most platforms have a strict method of specifying the condition. For example, in an "if" statement, the condition must be placed within parentheses. Unlike in

the case of Java platforms and in PHP, in ASP using VBScript the condition can simply be written after the "if" keyword without the need for enclosing it in parentheses. This feature makes ASP simpler for students to write.

Finally, control structures usually involve some block of code that will be executed. Clearly identifying such a block of code is a great aid to readability, especially for novice programmers. As mentioned earlier, Java uses C-type braces (**{** and **}**) to group statements together which makes it difficult to determine which group is being ended when an "**}**" appears. ASP's control structure ends with one or more keywords, e.g., End If, Loop, etc. (Deitel *et al.*, 2001, p.731) and not with { } as in the Java platforms.

There are two methods of forming compound statements or statement groups in PHP. Like Java, it uses C-type braces **{** and **}**). However, PHP offers an alternative syntax for some of its control structures. It makes things clearer by using a distinct closing syntax for each type of statement group; namely, *if*, *while*, *for*, *foreach*, and *switch* with the closing brace of *endif*, *endwhile*, *endfor*, *endforeach*, and *endswitch,* respectively (Bakken *et al.*, 2002b, p.62).

5.3.5 Functions and modularizations

Table 5.5 below gives the scoring for the platforms based on the criteria on functions and modularization. From the literature review and from programming experience, a summary of the results follows.

Prototypes serve as a blueprint for the creation of tangible objects in real life. A very useful concept in C++ is that one of the first steps is defining the function prototype (Deitel & Deitel, 2003, p.174-175). This approach has the advantage of preparing the mind of the programmer / student and enabling him or her to think about the function in terms of its return type, name to be assigned and the type and number of parameters it will receive. This is a good method of abstracting the problem and ensuring its adequate understanding and easy decomposition. Evaluating the platforms in line with this thought, it was observed that none of them require such early declaration of the prototypes.

Table 5-5. Scoring for the platforms based on the criteria on functions and modularization.

	Criteria questions	Servlet	JSP	ASP	PHP
1	Is it required to define prototypes for functions first before they are referenced later in the program?	1	1	1	1
2	Is it possible to independently compile or interpret and execute separate functional components of the program?	3	1	1	1
3	Are there many high level built-in functions, e.g. for sorting, printing etc.?	2	2	2	3
4	Is function overloading supported such that the same name can be used for the same functions that only differ in type and number of parameters?	3	3	1	1
5	Are there high level functions for sorting?	1	1	1	3
6	Is global variable prevented from overriding local variables?	3	3	3	3
	TOTAL	13	11	9	12

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

An old adage says “Rome was not built in a day”. Software programs are also developed incrementally and, in most cases, by different people in a team. It is thus important that separate functional components be compiled, executed and well tested. Avoiding undue access to the source code of programs being tested and used by other people is also crucial. Considering the four platforms, only Java Servlets allows independent compilation such that only the object code need be tested or used by others.

Given that most tangible objects in life are made up of smaller independent components, it is also important that programs make use of inbuilt functions instead of writing everything from scratch. While Java Servlets, JSP and ASP have many inbuilt functions, PHP has, in addition, functions that are not available in other programs. These include functions for slicing and splitting an array into chunks (Lerdorf & Tatroe, 2002, p.121) and iterating over arrays (Lerdorf & Tatroe, 2002, p.126), among others.

Unlike the case of operator overloading discussed earlier, function overloading is desirable for students because it allows using the same name for related functions distinguished by their inputs or signatures (Deitel & Deitel, 2003, p.219). Function overloading is supported in the Java-based platforms (Wigglesworth, 2000, p.131). We tested function overloading in ASP and it gave an error explaining that redefinition of the

function is disallowed. Also, according to DevDaily (1998), one can not re-declare a function or overload a function in PHP in the sense of C++ or Java's overloading.

Sorting arrays involve many steps that can be implemented in most programming languages. However, given that the emphasis in this study is on Web application development, it will save time if sorting can simply be accomplished using inbuilt functions. There are no array-sorting functions in the Java-based platforms (Wigglesworth, 2000, p.126). The sort method is available in ASP (Deitel *et al.*, 2001, p.719) and Lerdorf and Tatroe (2002, p.130) state that there are various array sorting functions such as sort ( ), ksort( ), rsort ( ), usort( ), arsort ( ), uasort ( ), and uksort ( ) in PHP.

It is very important that functions and the main programs interact effectively without corrupting their values. One way of ensuring this is by preventing global variables from overriding local variables. This is supported in the four platforms evaluated (Deitel & Deitel, 2003, p.739; Lerdorf & Tatroe, 2002, p.33).

5.3.6 Summary of results for the basic concepts criteria

As part of evaluating their suitability for teaching the concepts of Web application development, we have presented in this section the evaluation of the platforms according to the criteria set for ensuring suitability for teaching basic concepts. The summary of the results is given in Table 5.6 below.

Table 5-6. Total scores for the platforms based on the basic concepts criteria

	Criteria categories	Servlets	JSP	ASP	PHP
	Lexical Structures	19	21	28	24
	Data types	13	13	22	27
	Expressions and Operators	18	18	25	26
	Control Structures	11	11	23	19
	Functions and Modularizations	13	11	9	12
	TOTAL SCORE	74	74	107	108

The results in table 5.6 show that Java Servlet and JSP received an equal overall score, which is relatively lower than those of ASP and PHP. Similarly, ASP and PHP have an almost equal score, though PHP's score is slightly higher.

The results of the evaluation based on basic concepts criteria have been presented and discussed separately and in detail in this section as the basic concepts serve as a solid foundation for other Web application development concepts. In continuing with the results on the evaluation to ensure suitability of platforms for teaching Web applications, the next section presents the results of the evaluation of the platforms based on the criteria set for other concepts that are important for teaching Web application development in tertiary institutions.

5.4 Evaluating the suitability for teaching objects-oriented concepts

Table 5.7 below reflects the actual scoring for the middleware platforms based on the responses on object-oriented principles. From the review of various literature and from programming experience, a summary of the results using the criteria on object-oriented concepts for the four platforms evaluated follows.

Wigglesworth (2000, p.199) states that *classes* are the building blocks of object-oriented programs. In order to model real life objects, they are grouped together into *classes*. In the object-oriented paradigm, a *class* is a collection of variables and functions working with these variables (Bakken *et al.*, 2002b, p.70). Programmers can create an object of a class and can execute methods on the object (Deitel & Deitel, 2003, p.406). The Java platform is entirely class-based in that it forces one to put all code in classes (Wigglesworth, 2000, p.148). Even a complete program is a *class* structure (Wigglesworth, 2000, p.202). In fact, only the comments, package statements, and import statements can appear outside a class definition (Wigglesworth, 2000, p.151).

ASP supports the *class abstraction* as a form of the object-oriented paradigm. Objects encapsulate (i.e. wrap together) data (attributes) and methods (behaviours); this ensures

that objects have the property of information hiding. In ASP using VBScript, the unit of object-oriented programming is the *class* (Deitel *et al.*, 2001, p.754). Early versions of ASP using VBScript did not allow programmers to create their own classes, but present versions allow VBScript programmers to craft new *classes* and reuse existing ones (Deitel *et al.*, 2001, p.755). PHP also supports the *class abstraction* as a form of the object-oriented paradigm (Bakken *et al.*, 2002b, p.70).

Table 5-7. Scoring for the platforms based on the criteria on object-oriented concepts.

	Criteria questions	Servlet	JSP	ASP	PHP
1	Is there support for the class definition demonstrating the encapsulation of data with methods?	3	3	3	3
2	Is there support for constructor functions, including the overloading of the constructors?	3	3	1	3
3	Is there any support for information hiding?	3	3	3	3
4	Is it possible to independently compile or interpret and execute separate classes so as to build the program incrementally?	3	3	3	3
5	Is there support for classes within classes, illustrating hierarchical nature of classes?	3	3	3	3
6	Is there support for inheritance, illustrating inheritance relationships?	3	3	1	3
7	Is there support for abstraction with facilities for overriding abstracted features?	3	3	1	3
8	Is polymorphism supported?	3	3	1	3
9	Are there facilities for checking whether an identifier is an object or knowing the class to which an object belongs?	3	3	3	3
10	Is it possible to use pseudo-variables such as “this” in order to access its own functions and variables from within a class?	3	3	1	3
11	Is it possible to use special names such as “parent” or “super” to refer to the name of the base class instead of using the literal name of the base class in the code?	3	3	1	3
	TOTAL	33	33	21	33

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

A *constructor* is a special method that has the same name as a class and initializes the objects of that class (Wigglesworth, 2000, p.156). An advantage of *constructors* is that they can be overloaded to create different objects with different inputs. Wigglesworth (2000, p.168) remarks that constructor overloading provides alternative methods of

creating an object. One may require a default constructor, a constructor that sets initial values, a constructor that copies another object or that converts from one type to another.

The Java-based platforms support the use of constructors. They provide default constructors if programmer does not define any constructors for a class (Wigglesworth, 2000, p.160). ASP does not support the use of constructors (Deitel *et al.*, 2001, p.754-762). PHP's *classes* can have constructors (Lerdorf & Tatroe, 2002, p.142; Ross *et al.*, 2000).

Classes have the property of information hiding such that while class objects may know how to communicate with others, they are normally not allowed to know how other classes are implemented (Deitel & Deitel, 2003, p.406). The goal of information hiding is to protect the code from interferences. Methods can be called without knowledge of their implementation. It encapsulates properties by declaring them *private* and provides *public* accessor (*getter/setter*) methods for reading and modifying their values (jGuru, 2000). Deitel *et al.* (2001, p.754) add that Information hiding is crucial to good software engineering.

To achieve information hiding, the Java-based platforms use the *public, private, protected* or the default *package* access specifiers (Van Hoff, 1997; Wigglesworth, 2000, p.153). Using ASP's VBScript, information hiding is achieved via the methods collectively referred to as properties (Deitel *et al.*, 2001, p.757). On the other hand, PHP does not have the concept of *private* and *public* methods or properties (Lerdorf & Tatroe, 2002, p.142). Yager (2001) adds that one of the few roles PHP, aided by its many extensions, cannot fulfill is its lack of *private data members* and *private function members*, among other things, limiting PHP's usefulness in complex projects.

As in the case of functions, it is important to be able to compile or interpret and execute separate classes independently so as to build the program incrementally. While only Java Servlets allow independent compilation, other platforms allow independent translation, such that students can see the effects of their incremental builds.

The concept of classes within classes is included in the criteria for the sake of illustrating the hierarchical nature of classes. As all the platforms support the use of classes, it is simply a matter of putting one class within another.

Wigglesworth (2000, p.199) states that much of the power of object-oriented programming derives from the ability to model *data types* that are related by inheritance. Inheritance makes it possible to replace existing methods (inherited from the super-class) with ones that are more suited to the derived class. This allows for reuse of code.

The Java-based platforms fully support inheritance (Wigglesworth, 2000, p.202) to the extent that every Java class is a direct subclass of the core class *Object*. In ASP using VBScript, inheritance is not supported, though it can be simulated (Press, 2003). PHP however, like Java, implements inheritance using the "extends" keyword. They also both allow a derived class method to override the same method defined in the parent class (Lerdorf & Tatroe, 2002, p.145).

Inheritance is the concept behind polymorphism as (Deitel & Deitel, 1998, p.566) explains that polymorphism allows objects related by inheritance to respond differently to the same message, thereby ensuring extensibility. Only the Java-based platforms and PHP, which support inheritance, can implement polymorphism; ASP cannot.

To avoid confusion, students will often need to check the class of an identifier. The Java-based platforms use the getClass() function to check the class of an identifier (Wigglesworth, 2000, p.117). ASP using VBScript has a function named *IsObject* to check whether the *variant data type* is an object, as well as a function named VarType() to determine the subtype of the identifier (Deitel *et al.*, 2001, 735). PHP has functions to check whether an identifier is an object and to determine the class to which the object belongs (Lerdorf & Tatroe, 2002, p.149).

It is sometimes useful to use a pseudo-variable to access the properties of the current object and to call other methods on that object. The Java-based platforms use the "this" keyword for such reference (Wigglesworth, 2000, p.153) in a similar way to PHP (Lerdorf & Tatroe, 2002, p.144). Inability to find such a keyword in major ASP texts suggests that ASP does not support such feature.

It is also useful to use special names such as "parent" or "super" to refer to the name of the base class instead of using its literal name in the code. It reduces confusion and may also shorten the code. The Java-based platforms use the "super" keyword to refer to the super class for the current object (Wigglesworth, 2000, p.216) while PHP uses the "parent::" notation for such purposes (Lerdorf & Tatroe, 2002, p.145). Since ASP does not support inheritance, it doesn't have such notation.

5.5 Criteria for evaluating the suitability for teaching the Web techniques

One way in which Web development platforms differ from the general programming languages is in their support for Web development techniques used for the generation of both static and dynamic Web contents. Table 5.8 below gives the scoring for the platforms based on the criteria on Web techniques. A summary of the results using formed from the review of various literature and from programming experience for the four platforms evaluated follows.

From table 5.8, most of the Web techniques features are supported by all the platforms, possibly because that is the major goal of all platforms. In order to serve static information that accompanies dynamic contents, the associated Web servers all have facilities for static Web page serving.

The HTTP headers serve to pass valuable information, including error status, along with the message body and as such all the platforms have facilities to extract information from the HTTP request-headers, and to embed or set information into the HTTP response headers (Lerdorf & Tatroe, 2002, p.175-176). Possibly because they are all Web-based,

we note that all the platforms have methods of obtaining relevant information about both client and server from the server.

Table 5-8. Scoring for the platforms based on the criteria on Web techniques.

	Criteria questions	Servlet	JSP	ASP	PHP
1	Are there facilities for static Web page serving by the associated server?	3	3	3	3
2	Are there facilities to extract information from the HTTP request headers?	3	3	3	3
3	Are there facilities to embed or set information into the HTTP response headers?	3	3	3	3
4	Are there ways of obtaining relevant information from the server?	3	3	3	3
5	Is there availability of default folders for placing program code?	3	3	3	3
6	Is there the possibility of sending information via a service method that could implement both GET and POST methods?	3	3	3	3
7	Do numeric values accepted from a Web form retain their numeric type instead of being in string format which still has to be reconverted ?	1	1	3	3
8	Are there simple facilities or functions for form validation, e.g. checking non-empty fields, range of values, content of fields etc.?	2	2	3	2
9	Are there facilities for identifying users in a session, e.g. using cookies, hidden variables or URL rewriting?	3	3	3	3
10	Is there the possibility of setting the age of cookies thereby making cookies last longer?	3	3	3	3
11	Are there facilities for maintaining application objects so as to maintain global application states?	2	2	3	2
	TOTAL	29	29	33	31

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

Novice Web developers would prefer putting their code in default folders rather than creating and mapping new folders. All the platforms have default folders for this purpose.

Experience shows that passing information via *GET* methods embeds the values along with the URL and as such is useful for debugging. But for good security, *POST* is more useful as it does not show the information on the URL. All the platforms allow sending of information via a service method that could implement both *GET* and *POST* methods.

When numeric values are accepted from a Web form, it is better that they retain their numeric type instead of being in *String* format which still has to be reconverted. This is true in all the platforms except the Java-based ones where form values that are not *Strings* must be parsed from *Strings* to other types (Hall, 1999, p.435) using methods such as *parseInt()* to convert back to integer (Wigglesworth, 2000, p.821). However, program testing shows that in the case of ASP, when the operator used on the variables is an addition operator "+", it always assumes the *String* concatenation operator "+", thereby giving results such as "2+5 = 25". ASP similarly provides the conversion functions such as *CInt()* to explicitly convert to integer (Deitel *et al.*, 2001, 741) in such situations.

The need for the explicit conversion explained above which arose due to confusion between addition and *String* concatenation is not necessarily applicable in the case of other arithmetic operators such as subtraction "-", multiplication "*" and division "/". This makes such problems difficult to identify even for experienced programmers. One can then imagine how long it will take students identify such problems.

Validation of values accepted from Web forms is important to ensure the integrity of processing. In the Java-based platforms, one can use the *length()* method to determine whether the field is empty and one can "catch the NumberFormatException" if the entered value is non-numeric (Wigglesworth, 2000, p.821). ASP has simple functions such as *IsDate(), IsNumeric(), and IsEmpty()* to assist in the validation (Deitel *et al.*, 2001, 735). Also, there are facilities or functions for form validation such as checking non-empty fields, range of values, content of fields *etc*. in the PHP platform (Lerdorf & Tatroe, 2002, p173), but they are not as simple as in ASP.

The effect of HTTP being a stateless protocol is that it does not retain details of transactions (Lerdorf & Tatroe, 2002, p.178; Wigglesworth, 2000, p.827). All the platforms have facilities for identifying users in a session, for instance using *cookies*, hidden variables or URL rewriting, and there is the possibility for setting the age of *cookies* thereby making them last longer (Lerdorf & Tatroe, 2002, p.178-179). In addition, all the platforms have ways of maintaining global *Application* states.

5.6 Evaluating the suitability for teaching the database handling capabilities

Table 5.9 below gives the scoring for the platforms based on the criteria on database handling capabilities. Below also is a summary of the results on database handling capabilities for the four platforms evaluated, formed from the review of various literature and from programming experience.

It is very important for the Web platforms to support various Database Management Systems (DBMS) so that Web applications developers can choose the one most appropriate for their purposes and also to leverage the knowledge of any DBMS the students may already have. Experience shows that the Java-based platforms support various database systems. ASP, however, being Microsoft based, supports a limited number of database systems. According to Lerdorf and Tatroe (2002, p.189), PHP supports over 20 databases, including the most popular and open source varieties.

Table 5-9. Scoring for the platforms based on the criteria on database handling capabilities.

	Criteria questions	Servlet	JSP	ASP	PHP
1	Does the platform support various database management systems so as to leverage the knowledge of any DBMS the students may already have?	3	3	1	3
2	Is the DBMS sometimes included in the installation package so as to ease the installation configuration settings?	1	1	2	3
3	Does the platform support MsAccess as the common DBMS via both odbc and jdbc?	3	3	3	3
4	Are there user-friendly facilities for loading drivers, opening and closing connections, retrieving database items etc.?	1	1	3	2
5	Are there possibilities for retrieving data items in their original types rather than converting back from strings?	1	1	3	3
6	Are there facilities for navigating the fetched datasets?	2	2	3	2
	TOTAL	11	11	15	16

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

Students often have problems configuring the interaction between the Web server and the database. It would be very helpful if the DBMS was included in the installation package so as to ease the choice problem and the installation configuration settings. The Java-based platforms do not include any database along with their installation package. ASP is Microsoft based and so the natural choice of database is Microsoft Access. This may also ease configuration settings between Microsoft XP/2000 and Microsoft Office as they are installed. PHP, unlike others, is sometimes bundled with MySQL. Being "bundled" together also increases the degree of interoperability between PHP and MySQL, as they are automatically installed and configured (Bakken *et al.*, 2002a, p.5).

Moreover, since the most common operating system used by our students is Microsoft based, support for Microsoft Access is important. The Java-based platforms support the use of Microsoft Access. As for ASP, Microsoft Access is strongly supported as both have been developed by Microsoft. PHP supports Microsoft Access as DBMS via both ODBC and JDBC (Bakken *et al.*, 2002a, p.4).

We noted that Microsoft, in line with its goal of ensuring good Graphical User Interface, and unlike the Java-based platforms, has introduced user-friendly facilities for establishing the interaction between the Web server and the DBMS. Active Server Pages communicate with databases through ADO (ActiveX Data Objects), which provide a uniform way for a program to connect with a variety of databases in a general manner without having to deal with the specifics of those database systems (Deitel *et al.*, 2001, p.810). PHP also has features that ease interaction with MySQL as the database.

Attempting to perform arithmetic calculations with values retrieved from database in *String* form will frustrate many students. In the Java-based platforms, retrieved numeric data items are in *String* form and must be converted to numeric. In ASP and PHP, however, there is the facility to retrieve data items in their original types rather than converting them back from *Strings*.

Experience shows that results of retrieval operations are stored in datasets and one needs to navigate or move in steps through the datasets to access the data, records by records. Unlike others relying on loops to move through datasets, ASP has facilities for navigating the fetched datasets. These include moving to the top and bottom of the dataset or using the MoveNext() method by incrementing the record set pointer to point to the next record (Deitel *et al.*, 2001, p.816).

5.7 Evaluating the suitability for teaching the file processing capabilities

Table 5.10 below provides the actual scoring for the four platforms based on the responses using the criteria on file processing capabilities. From the review of relevant literature and from programming experience, a summary of the results on file processing capabilities for the four dynamic Web platforms evaluated is given below.

Table 5-10. Scoring for the platforms based on the criteria on file processing capabilities.

	Criteria questions	Servlet	JSP	ASP	PHP
1	Are there commands or functions for creating, opening and closing files on the server's disks?	3	3	3	3
2	Are there facilities for checking the existence of the files on disk?	3	3	3	3
3	Are there facilities for writing or storing records sequentially on the files?	3	3	3	3
4	Are there facilities for reading records sequentially from the files?	3	3	3	3
5	Is there a file lock facility to prevent more than one script from accessing a file at the same time?	2	2	2	3
6	Are there facilities for uploading files from users' disks?	3	3	3	3
7	Are there facilities to contain information about the uploaded files?	2	2	2	3
8	Can the maximum size of the uploaded files be set?	2	2	2	3
9	Are there facilities to check whether a file was successfully uploaded?	2	2	2	3
10	Are there facilities for moving uploaded files from the server's default directory to desired directories?	2	2	2	3
	TOTAL	25	25	25	30

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

Without the commands or functions for creating files on the server's disks and the facilities for storing records sequentially on the files, there could be no file processing. The Java-based platforms feature commands for creating files, storing records sequentially, and reading records from files. These are achieved through the "FileInputStream" class and the "ObjectInputStream" constructor (Deitel and Deitel, 2002, p.497). ASP uses the File System Objects to provide the ability to manipulate files, directories and drives (Deitel *et al.*, 2001, p.785). PHP uses the *fopen()* command to open and read files (Deitel and Deitel, 2002, p.497). The *fread()* and *fwrite()* functions are used to read from and write to file respectively (Deitel and Deitel, 2002, p.498-499).

It is necessary to determine the existence of a file to avoid the problem of a program failing because a specified file does not exist, and to avoid overwriting existing files. The Java-based platforms use the *exists()* method for this purpose (Wigglesworth, 2000, p.327). ASP and PHP use the *file_exists* function for this purpose (Deitel *et al.*, 2001, p.785; Deitel and Deitel, 2002, p.498).

For the purpose of reliability, it is better to allow only one script to access a file at the same time. Only PHP has a file locking facility (Deitel and Deitel, 2002, p.500).

As users send information to the server, they should also be able to upload files. The Java-based platforms can process the uploading of files via the HTTP request by instantiating a *ServletInputStream* using the *getInputStream* method of the request object (Kurniawan, 2001). ASP also supports file uploading (Khan, 2003). In addition, PHP has facilities for uploading files from users' disks (Lerdorf & Tatroe, 2002, p.172).

PHP also have facilities to contain information about the uploaded files, setting the maximum size for these to avoid denial of service attack. Only PHP explicitly specifies how to set the maximum size for the uploaded files (Lerdorf & Tatroe, 2002, p.290).

For purposes of reliability there should be facilities to check whether a file has been successfully uploaded. Unlike other platforms, only PHP has such a facility (Lerdorf &

Tatroe, 2002, p.172-173). Other platforms rely on the use of program manipulations to achieve this.

Platforms should also have the facility to move uploaded files from the server's default directory to other desired directories. The Java-based platforms, ASP, as well as PHP allow the setting of the specific path where the uploaded file should be saved on the server (Khan, 2003, p.2; Kurniawan, 2001; Lerdorf & Tatroe, 2002, p.172-173).

5.8 Evaluating the suitability for teaching exception handling in Web applications

Table 5.11 below presents the scoring for the platforms based on the criteria on exception handling capabilities. Formed from the review of various literature and from programming experience, a summary of the results on database handling capabilities for the four platforms is also presented below.

Table 5-11. Scoring for the platforms based on the criteria on exception handling capabilities.

	Criteria questions	Servlet	JSP	ASP	PHP
1	Is it possible to set and extract the status code representing error status stored by the server?	3	3	3	3
2	Does the platform translate error code and display pages that will present the errors in a meaningful language?	2	2	3	3
3	Are there facilities for specifying the errors that the program is allowed to leave unhandled?	3	3	2	3
4	Are there facilities to disable or suppress the errors messages for a single expression?	2	2	2	3
5	Are there facilities to disable or turn off error reporting completely?	1	1	1	3
6	Are there facilities for intercepting and handling run-time errors in the programs such that the program can recover from these errors?	3	3	3	3
7	Are there facilities for exiting gracefully without abruptly stopping the program, thereby ensuring consistency of program states?	3	3	3	3
8	Are there facilities for logging or recording errors on files on disks?	2	2	2	3
	TOTAL	19	19	19	24

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

An HTTP response sent from the Web server to browsers includes a status code which indicates whether the request was successful (the 200 status code) or unsuccessful otherwise (Sun Educational services, 2002, Module 9, p.3). All the platforms have facilities to extract information from the HTTP request-headers, and to embed or set information into the HTTP response headers. The Java-based platforms use a series of "set" methods for this purpose (Deitel and Deitel, 2002, p.429-455) while ASP uses the "AddHeader" method (Chapkhanov, 1999, Chapter 12, p.11) and PHP uses the *header ()* function (Lerdorf & Tatroe, 2002, p.175-176).

The status code is sometimes meaningless or even confusing to the programmer (Deitel *et al.*, 2001, p.646). Anyone who has used the Java-based platforms will confirm that they provide terrifying error diagnostics. The author notes that ASP and PHP translate status code and display pages that will present the errors in a more meaningful message.

To speed up testing, it is sometimes necessary to specify the errors that the program is allowed to leave unhandled. In the Java-based platforms, one can declare uncaught exceptions by listing them in the ***throws clause*** (Wigglesworth, 2000, p.248). ASP does not allow errors to be left unhandled (Deitel *et al.*, 2001, p.646), except when the author specifically "comments out" these statements to avoid their execution. PHP allows listing the error conditions that are caught and also has facilities to disable or suppress the error messages for a single expression (Lerdorf & Tatroe, 2002, p.304).

In other situations, it is better to disable or turn off error reporting completely. Only PHP allows one to turn off error reporting entirely (Lerdorf & Tatroe, 2002, p.305). This ensures that, regardless of the errors encountered while processing and executing script, no errors will be sent to the client except parse errors which cannot be suppressed (Lerdorf & Tatroe, 2002, p.305).

For better control of errors than simply hiding them, error handlers can be used. The error handler is called when a condition of any kind is encountered. It can do anything one requires, from logging to a file to printing error messages. Using the Java-based

platforms, Wigglesworth (2000, p.253) lists some of these methods which include "trying" code and "catching" exceptions by encapsulating them in ***try*** blocks.

In order to handle errors more elegantly in ASP using VBScript, one can use the *ONERROR* event to launch error-handling code when an *ONERROR* event is triggered. This can be used to write error messages to the status bar of the browser (Deitel *et al.*, 2001, p.646). As explained by Lerdorf & Tatroe (2002, p.305), the basic process in PHP creates an error-handling function and registers it with the *set_error_handler( )* method. This intercepts and handles run-time errors in the programs such that the program can recover from the errors.

Sometimes, it is necessary to exit gracefully without stopping the program abruptly, thereby ensuring consistency of program states. The Java-based platforms use the *System.exit()* function to circumvent displaying the error message because the program ends before the error occurs, thus ending the application gracefully (Wigglesworth, 2000, p.675). In ASP, the *exit* function exits the program (Deitel *et al.*, 2001, p.1169). In PHP, the *exit ()* function which is also an alias for *die ()* can be used to print out messages before ending the execution of a script when necessary (Lerdorf & Tatroe (2002, p.54).

For further reference and perusal, it is necessary to log or record errors on files on disks. This is typically done by the associating Web server, and most Web servers used with the platforms have such facility. However, in addition to this and unlike other platforms, PHP provides a built-in function, *error_log()*, to log errors to files and even to email addresses (Lerdorf & Tatroe, 2002, p.306).

5.9 Evaluating the suitability for teaching the XML Support capabilities

Table 5.12 below provides the scoring for the platforms based on the criteria on XML support capabilities. From programming experience, the author notes that as in the case of Web techniques, all the platforms adequately support XML handling, possibly due to the importance of XML in intersystem communications. Moreover, XML documents are

created just as HTML documents. Also, most other features that enhance the use of XML, such as using style sheets, are linked to HTML rather than to the platforms.

Table 5-12. Scoring for the platforms based on the criteria on XML support capabilities.

	Criteria questions	Servlet	JSP	ASP	PHP
1	Is there a possibility of delivering XML documents rather than raw HTML files to the browser?	3	3	3	3
2	Is there support for attaching style sheets to the XML data?	3	3	3	3
3	Are there facilities for embedding an XML data island directly in an HTML page by using the XML element?	3	3	3	3
4	Is it possible to reference an XML data island in an external file from HTML using the SRC attribute?	3	3	3	3
5	Are there facilities for binding an XML data island to an HTML table by setting the DATAFLD attribute for each column (“TD”) in the table?	3	3	3	3
	TOTAL	15	15	15	15

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

5.10 Summary and conclusions

In this chapter we have presented the results on the evaluation of the platforms based on the criteria that will ensure suitability of the platform for teaching the Web application development concepts. A summary of the results is presented in table 5.13 below. From the results in the table, it is evident that ASP is not as object oriented as the other platforms. This is probably one of the reasons for the advent of ASP.NET, aimed at putting ASP on the same level as other object-oriented platforms.

With respect to Web techniques, all the platforms are considered suitable, possibly because that is their primary objective. However, the simplicity of ASP and IIS places ASP slightly ahead of the others according to the criteria for these Web techniques.

According to the criteria for file processing and exception handling capabilities, PHP scores higher than all the other platforms. For XML support, all the platforms are rated equally because they all produce XML documents in the same way as HTML documents

are produced. Moreover, other supports for XML are dependent on the browser and not on the platform.

Table 5-13. Total scores for the platforms based on the criteria on the suitability for teaching Web application development concepts.

	Criteria categories	Servlets	JSP	ASP	PHP
	Basic concepts	74	74	107	108
	Object-oriented programming	33	33	21	33
	Web techniques	29	29	33	31
	Data base handling capabilities	11	11	15	16
	File processing capabilities	25	25	25	30
	Exception handling capabilities	19	19	19	24
	XML support	15	15	15	15
	TOTAL SCORE	206	206	235	257

In conclusion therefore, the results show that PHP is considered the most suitable for teaching the Web application development concepts. The next chapter presents the results of the evaluation of the dynamic Web platforms using the criteria to satisfy other constraints. These include the results from the evaluation of the platforms based on the criteria set to ensure the ease of use, high latency, affordability and portability of the platforms.

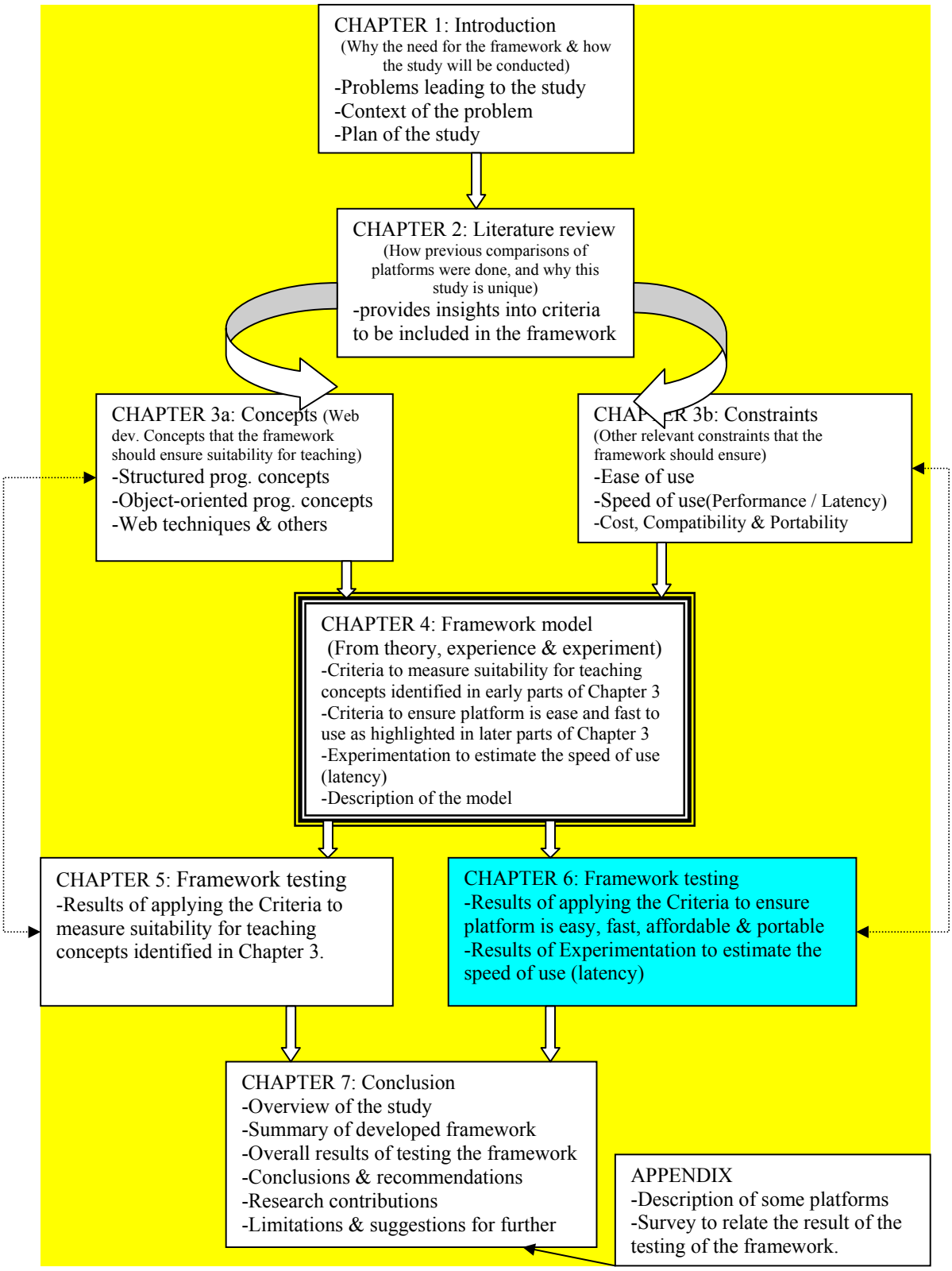

Figure 6-0. Dissertation map highlighting the relativity of chapter 6 to the rest of the dissertation.

CHAPTER 6

6 RESULTS FOR THE SUITABILITY OF THE WEB PLATFORMS USING THE EASE OF USE, LATENCY, COST AND PORTABILITY CRITERIA

6.1 Introduction

In the last chapter, we presented the results of the evaluation of the platforms based on various criteria that would ensure the suitability for teaching the concepts of Web application development. That will ensure that in the first instance, the platform is suitable for doing what it is meant for. However, in addition to that, it is important to ensure that the various tasks in Web application development are done with adequate ease and in reasonable time. Moreover, it is also important to ensure affordability and portability of the platform, so that students can be able to practice at home what they have learnt and done in the practical classes, and yet, ensure that the expertise gained will be very valuable in their future works.

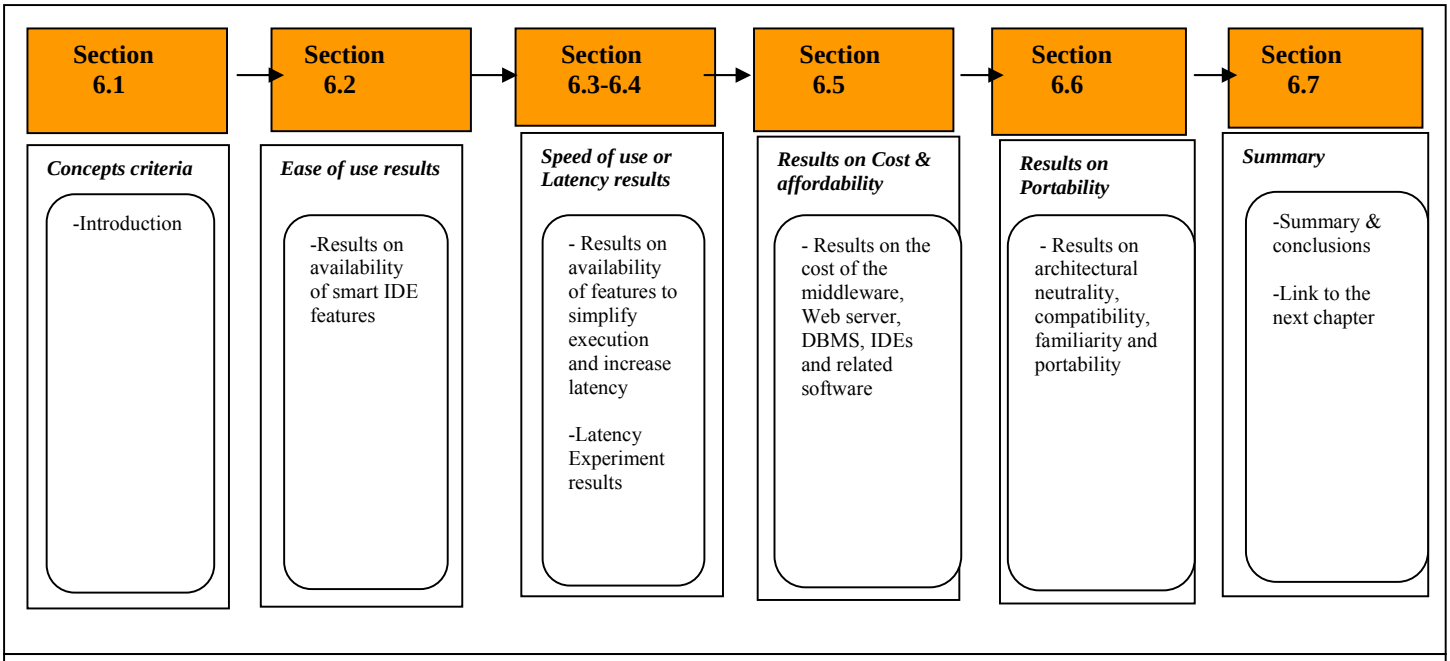

Figure 6-1. Pictorial outline of chapter 6.

This chapter presents the results of the evaluation of the Web platforms against the criteria set to evaluate their ease of use, latency, affordability and portability. Figure 6.1 above gives a pictorial outline of the chapter. Section 6.1 introduces the chapter and section 6.2 looks at the criteria to ensure the availability of desirable features for a smart Integrated Development Environment for the platforms.

Section 6.3 considers features that will simplify execution and decrease the latency (speed of execution) of programs, supplemented with the results of the latency experiment in section 6.4 to rank the latency of the platforms. This is followed by section 6.5 that presents the results on the affordability of the platforms, and section 6.6 on the architectural neutrality and portability of the platforms. Lastly, a summary of the chapter is presented in section 6.7.

When asking the questions serving as some of the criteria of the framework later, we use a close-ended "Yes/No" questions, on a scale of **1 to 3;**

where 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

To answer the questions when evaluating the platforms, we used referenced information from textbooks, journals, authoritative web sites, as well as using author's observation and experiences. However, users of the framework can use their own references and experiences, especially if evaluating platforms different from the four that we selected in section 5.2 of chapter 5 namely Java Servlets, Java Server Pages (JSP), Active Server Pages (ASP) and PHP Hypertext Processor.

6.2 Availability of features of smart Integrated Development Environment

Table 6.1 and table 6.2 below show the scoring for the dynamic Web platforms based on the criteria on the availability of features of smart IDEs. A summary of these results based on a review of various literature, detailed observation and testing of various IDEs and from programming experience is given below.

The use of an interpreter rather than a compiler will allow students to instantly see results of programs up to what has been coded so far, irrespective of whether the program is complete or not. The author notes that apart from Java Servlet, in which one needs to compile the program, other platforms use interpreters such that the incremental building of the programs becomes easier for students. In the case of JSP, however, the interpretation process involves internal compilation of the programs (Sun Microsystems, 1997).

Table 6-1a. Scoring for the platforms based on the criteria on the availability of features of smart IDEs.

	Criteria questions	Servlet	JSP	ASP	PHP
1	Does the platform use interpreters rather than compilers?	1	2	3	3
2	Is the IDE both DOS and Window based?	3	1	1	3
3	Does the IDE automatically create directories to store the project?	3	3	3	1
4	Does the IDE show the directory tree for the project?	3	3	3	3
5	Does the IDE allow the menu-driven addition of Web items such as HTML pages, scripts, style sheets, etc.?	3	3	3	3
6	Does the IDE enhance maintenance by encouraging good readability by way of automatically formatting and indenting code?	3	3	2	3
7	Does the IDE have hotkey matching of braces, parenthesis, and angle brackets?	3	3	1	2
8	Does the IDE have customizable code formatting templates?	3	3	3	3
9	Is the IDE equipped with tracing and debugging aids to assist students in debugging their code?	3	3	3	3
10	Does the IDE offer other useful and smart editing facilities such as identifying errors as soon as commands are entered, rather than waiting till compilation time?	1	1	1	1
11	Does the IDE give syntax highlighting with configurable colours?	3	3	3	3
12	Does the IDE have the ability to switch case sensitivity on and off as necessary?	3	3	1	1
13	Does the IDE have function and variable name autocompletion?	3	3	1	2

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

Also, for prompt testing and incremental building of the programs, it is better that the platform allows execution from both the DOS prompt and the browser. The Java Servlet

allows execution from the DOS prompt through the use of the "Servlet runner" program (Sun Microsystems, 1997) and also from the browser. JSP and ASP are only executable via the browser. The PHP engine can be invoked as either a CGI-callable interpreter from supported UNIX platforms and Windows NT, or as an in-process Apache module that can be viewed from all HTML-capable browsers (Ross *et al.*, 2000). It can be used for server-side scripting via browser as well as command-line scripting via DOS prompt and Unix shell (Lerdorf & Tatroe, 2002, p.1).

Table 6-1b. Scoring for the platforms based on the criteria on the availability of features of smart IDEs continued.

	Criteria questions	Servlet	JSP	ASP	PHP
14	Does the IDE have pop-up parameter references for recognized functions?	3	3	3	3
15	Does the IDE have an HTML toolbox for adding HTML components as well as WYSIWIG editing?	1	1	3	1
16	Does the IDE enhance database integration by having database tools such as viewing tables, adding connection objects?	1	1	2	2
17	Is the IDE equipped with context sensitive helps to assist students who are "stuck" in certain programming activities?	3	3	2	1
18	Does the IDE allow searching documentation and API from the help menu?	3	3	3	1
19	Does the IDE show split views of long scripts?	1	1	1	3
20	Does the IDE show different views such as the explorer class view, program view, debugging view, output view?	3	3	3	2
21	Does the IDE have configurable keyboard shortcuts?	3	3	3	3
22	Does the IDE automatically construct the URL for the output?	1	1	3	3
23	Is the platform equipped with an independent GUI development tool?	3	3	2	1
24	Is there a class wizard to assist in the creating of classes?	3	3	1	1
25	Is it possible to start and stop the server from within the IDE?	1	1	1	3
26	Does the IDE support unlimited undo and redo?	3	3	3	3
	TOTAL	64	63	58	58

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

Students need to know in which directory to put the scripts, HTML pages and graphics. The JCreator IDE automatically creates directories to store the project for the Java-based

platforms (JCreator, 2004). The same goes for Microsoft Visual Interdev IDE for ASP (MSDN, 2004) and PhpEd for PHP (Yank, 2003). In addition, a look at all the IDEs shows that they allow menu-driven addition of Web items such as HTML pages, scripts and style sheets, which are put in the appropriate sub-directories.

Apart from creating the directory, the IDE should also show the directory tree for the project to enable the students to browse and modify files easily. This feature is available in JCreator for the Java-based platforms (JCreator, 2004), and in Microsoft Visual Interdev IDE for ASP (MSDN, 2004), as well as in PhpEd for PHP (Yank, 2003).

It is important that the IDE enhances maintenance by encouraging good readability by way of automatically formatting and indenting code. JCreator, which can be used as editor for Java Servlet and JSP programming, automatically formats and indents code (JCreator, 2004). Similarly, Microsoft Visual InterDev, usable as editor for ASP code, has formatting and indenting capabilities (MSDN, 2004). PhpED, which comes with PHP, also automatically formats and indents the code (Yank, 2003).

Since the parentheses and brackets are always in pairs, it is useful if the IDE has hotkey matching of braces, parentheses and angle brackets. Practical testing shows that the various IDEs such as JBuilder and JCreator, for the Java-based platforms, as well as the Visual InterDev for ASP have this facility. PhpEd for PHP also has this facility (Yank, 2003).

Since there is usually a special format for the HTML files, scripts and so on, it is preferable if the IDE has customizable code formatting templates such that students can just add contents to the templates. This feature is available in JCreator (JCreator, 2004) and in Microsoft Visual Interdev IDE for ASP (MSDN, 2004) as well as in PhpEd for PHP (Yank, 2003).

An IDE that is equipped with tracing and debugging aids will assist students in debugging their code. Examination of the various IDEs for the platforms shows that they all have tracing and debugging features.

It is important that the IDE should offer other useful and smart editing facilities such as identifying errors as soon as commands are entered, rather than waiting till compilation time. Examination of the various IDEs for the platforms shows that none have this feature.

Given that colours enhance visualization, it is of some importance that the IDE provides syntax highlighting with configurable colours. Practical testing of the various IDEs for the platforms shows that they all have this feature.

Another desirable feature is function and variable name auto-completion, since it will eliminate the use of wrong names and also speed up the coding process. The IDEs for the Java-based platforms have this feature. Unlike the IDE for ASP, PhpEd for PHP also has this feature (Yank, 2003).

Since the IDE knows the functions applicable in a platform, it will be useful if it has pop-up parameter references for recognized functions. The IDEs for the Java-based platforms, as well as Microsoft Visual InterDev for ASP, have this feature. PhpEd for PHP has it as well (Yank, 2003).

In accordance with the move towards GUI-based systems, it is better if the IDE has an HTML toolbox for adding HTML components as well as WYSIWIG editing. Unlike the IDEs for the other platforms evaluated, only the Microsoft Visual InterDev for ASP features this (MSDN, 2004).

Given that most Web applications involve the use of database, teaching will become easier if the IDE enhances database integration by having database tools such as those for viewing tables, adding connection objects etc. Detailed examination of the IDEs shows

that this feature is missing in the IDEs for the Java-based platforms, but available in Visual InterDev for ASP (MSDN, 2004), as well as in EasyPHP IDE for PHP. Fuecks (2004) confirms this by warning former VB users not to forget that there is ADOdb which is an ADO equivalent for PHP.

Furthermore, being equipped with context sensitive helps will enable such IDEs to assist students who are "stuck" in certain programming activities. Examination of the various IDEs shows that those for the Java-based platforms have such features, unlike the IDEs for the other platforms evaluated.

Detailed information will be easily accessed if the IDE allows searching documentation and API from the help menu. All the IDEs for the various platforms have such features (JBuilder, 2004; MSDN, 2004; Yank, 2003).

In long statements, it is useful if the IDE shows split views of the long scripts instead of requiring one to scroll and lose sight of beginning portions. Our observation was that this feature was missing in the IDEs for the Java-based platforms and ASP. PhpEd for PHP does, however, have this feature (Yank, 2003).

While coding and testing the developed system, one often needs to see different views such as the class view, program view, debugging view, and the output view. Our observations revealed that all the IDEs for the various platforms have this feature.

For those functions obtainable via the menu, it would be additionally beneficial if the IDE has configurable keyboard shortcuts. All the IDEs for the various platforms have this feature.

Even after the script is syntactically correct, students often have difficulty in constructing the applicable URL to view the results. A useful feature is when the IDE automatically constructs the URL for the output at the click of a button. Unlike the IDE for the Java-

based platforms, examination shows that this feature is available in the IDEs for ASP and PHP.

Although we have already established that it is useful for novices if the platform is case insensitive, it would be even better if the IDE had the ability to switch case sensitivity on and off as necessary. Observation shows that the IDEs for the Java-based platforms have such a feature (JCreator, 2004), unlike the IDEs for the other platforms under study.

If the platform is equipped with an independent GUI development tool, a situation where one has to learn and use another language for the front-end design can be avoided. The Java-based platforms have the Swing class for the front-end GUI design, unlike the IDEs for the other platforms that use HTML for the front-end GUI design.

Also most useful is a class wizard which will assist in the creation of classes. JBuilder for the Java-based platforms has such facility (DevDaily, 1998) but the IDEs for the other platforms do not. It is easier if one is able to start and stop the server from within the IDE. Only PhpEd for PHP has such facility. The IDEs for the other platforms do not.

Last but not the least, it is valuable if the IDE supports an unlimited undo and redo feature so that desired actions can be repeated and unwanted changes can be easily discarded. All the IDEs for the platforms being evaluated have undo and redo buttons (JBuilder, MSDN, 2004; Yank, 2003).

6.3 Evaluating features to simplify execution of programs

Table 6.2 below provides the scoring for the platforms based on the criteria on features to simplify execution and increase latency of programs. From the review of various literature and from programming experience, a summary of these results for the four platforms evaluated follows below.

Familiarity with a platform is very important in reducing the execution steps. Java Servlets and JSP programming build upon the introductory Java programming course

which students complete in their first year. Similarly, ASP, using VBScript, builds on the Visual Basic programming course completed in first year. PHP is, however, relatively new to students, although some of the syntax is similar to C++ and Java.

Table 6-2. Scoring for the platforms based on the criteria on features to simplify execution and decrease latency of programs.

	Criteria questions	Servlet	JSP	ASP	PHP
1	Does the platform integrate with other relevant components of the curriculum, such as the teaching of other programming languages?	3	3	3	2
2	Does the platform use relatively moderate disk space?	3	3	3	3
3	Does the platform have simpler execution steps, i.e. are there fewer settings to be made before it starts working?	1	1	3	2
4	Does the platform have simple deployment requirements, i.e. is the default directory on fewer levels relative to the "c:\" root directory?	1	1	3	3
5	Does the platform have a reasonable number of unique basic syntactical components such that a user can learn most commands off by heart?	1	1	3	1
6	Does the platform reduce the time spent on errors by pointing more directly to them?	1	2	3	3
7	Is there valuable help through standard documentation such that students can check out the correct syntax for statements?	3	3	2	3
8	Does the documentation have a good number of example programs pre-installed?	3	3	1	2
9	Does the platform generalize commands with consistency of commands such that features and functions are easily recalled by users?	3	3	3	3
10	Is the platform flexible by offering multi-threading?	3	3	1	1
11	Are there many higher-level commands or buttons to achieve some lengthy tasks?	1	1	3	3
12	Does the platform have simple architecture?	3	3	3	3
13	Does the platform have a garbage collection facility?	3	3	1	3
14	Is it possible to avoid the need for the program to be compiled first before execution?	1	2	3	3
15	Are the processes of compilation, linking and execution done behind the scenes by the compiler / interpreter?	1	3	3	3
16	Can programs be executed both from the command line and from the browser?	3	1	1	3
17	Are array bounds checked both at compile time and at runtime?	1	1	3	3
18	Does the platform feature high performance in database retrieval applications?	2	1	3	3
	TOTAL	37	38	45	47

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

Space taken up by the platform on the hard disk affects the speed of operation. This is one of the reasons why ASP.NET was not chosen as one of the platforms under study. Fortunately, all the platforms use a moderate amount of space.

Some platforms are easy to execute because they have simpler execution steps, such as having fewer settings to be made before it starts. When using Java Servlets, one has to set the *classpath* first. One also has to compile the programs whereas in the other platforms one simply copies the program into the appropriate directories.

In a similar context, the default directory for Java Servlet is on several levels relative to the "c:\" root directory. This is unlike other platforms that have simpler deployment requirements.

The number of unique basic syntactical components available in a platform may affect the recollection of the commands. Java Servlets, JSP and PHP feature large numbers of commands, unlike ASP using VBScript which has fewer commands, such that a user can easily remember most of these commands.

The time spent on errors can be reduced if the platform points more directly to them. Experience and practical testing show that Java Servlets and JSP list many code lines affected by just a single error. The other platforms are more direct.

Valuable help can be obtained through standard documentation that allows students to check the correct syntax for statements. The Java-based platforms have much official online documentation. The same applies to PHP, but ASP features less official online documentation. In the same context, the official online documentation that accompanies Java Servlets and JSP has a good number of example programs pre-installed, allowing students to follow and learn their coding examples.

Generalized commands and consistency of commands will reduce the time spent in executing the programs. All the platforms have window-based menus for executing programs such that execution commands are easily recalled by users.

Multi-threading is a form of flexibility offered by some platforms. Executing multiple threads simultaneously may reduce the execution time for programs. A search into various texts and websites showed that the Java-based platforms offer multi-threading. However, ASP and PHP do not directly support this feature.

Availability of higher-level commands or buttons to achieve some lengthy tasks will reduce the time spent executing programs. Unlike in Java Servlet that has to compile first, an execute menu choice will compile and execute the program in the other platforms.

The architecture of the platform may also affect the speed of execution (Menasce, 2003). All the platforms allow client-server-database architecture where the database is on the server. This is simpler for students to use, unlike EJB where the database is on another server. This is the reason why EJB has not been considered for evaluation.

The garbage collection facility removes data items that are no longer in use from memory and reclaims the resources held by such objects (Van Hoff, 1997; Wigglesworth, 2000, p.115), thereby freeing memory and leading to improved performance. According to Spolsky (2004), the really significant productivity advancement in programming has been from languages which manage memory automatically. The Java-based platforms utilize the garbage collector (Van Hoff, 1997; Wigglesworth, 2000, p.159). ASP does not, while PHP also has a garbage collection facility (Lerdorf & Tatroe, 2002, p.33).

Compilation and execution are completed separately in some platforms, thereby increasing the number of tasks needed before seeing results from the program. Experience shows that Java Servlets require compilation before executing the program; the other platforms do the compilation, linking and execution behind the scene.

To make testing faster, some platforms allow programs to be executed either via the command line or the browser. The Java Servlet allows execution via the command line using the Servlet runner (DevDaily, 1998). JSP and ASP do not. PHP, like Java Servlet, also allows execution either via the command line or the browser Lerdorf & Tatroe, 2002, p.33).

To speed up execution of programs, values are usually stored in arrays which are in memory instead of values in files. The speed of execution can be reduced if the array bounds are checked both at compile time and runtime. The Java-based platforms check array bounds at compile time and runtime, but since ASP and PHP do not compile the code, they cannot be expected to check array bounds at compilation time.

Most dynamic Web applications involve database retrieval. It is thus important that the platform has good performance in database retrieval applications so that they can retrieve results quickly. Further detail on the latency experiment is given in the next section.

6.4 Latency experiment results and discussions

In section 6.3 we presented the results on the evaluation of the platforms based on various criteria that would simplify execution and decrease latency of programs. This is supplemented by the latency experiment, the results of which are presented in this section. These are aimed at satisfying the constraint of suitability for the recurrent testing that takes place in the students' laboratory, by ensuring that the chosen platform will execute quickly.

Users of the developed framework can choose to conduct their own latency experiment provided this ranks the latency of the platforms being evaluated. Furthermore, users can allocate weight to the experiment relative to the other components of the framework. Given below are the results of the latency experiment presented at the *Informing Science + Information Technology Education (InSITE) 2005* conference in Arizona, USA, and published as a full paper in the conference associated Journal. Further details of this experiment in Dehinbo (2005) are available on http://insite.nu.

The results and the discussions are presented along with sample screen snapshots. Each of the four programs written was executed on the server via the browser. The results follow.

6.4.1 Sample output screen capture

Sample outputs from the programs are given below in Figure 6.2 to 6.5:

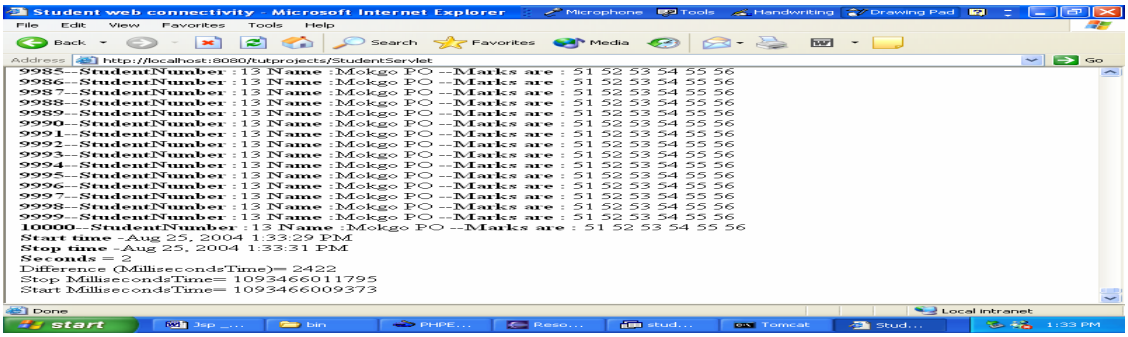

Figure 6-2. Database retrieval output using Java Servlet.

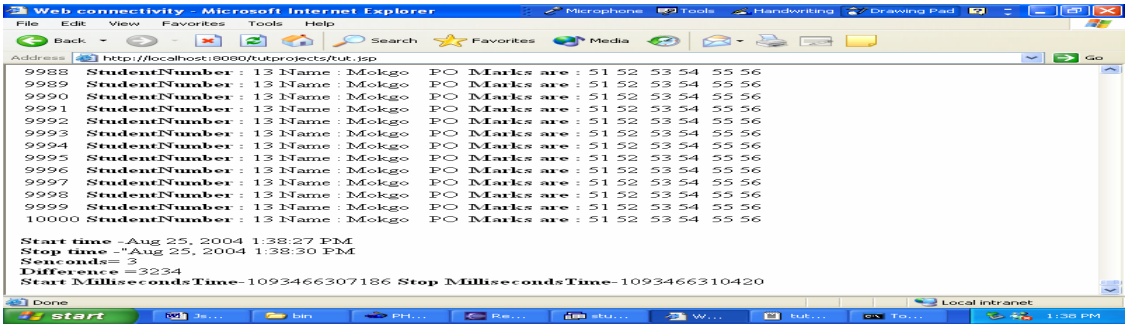

Figure 6-3. Database retrieval output using Java Server Pages.

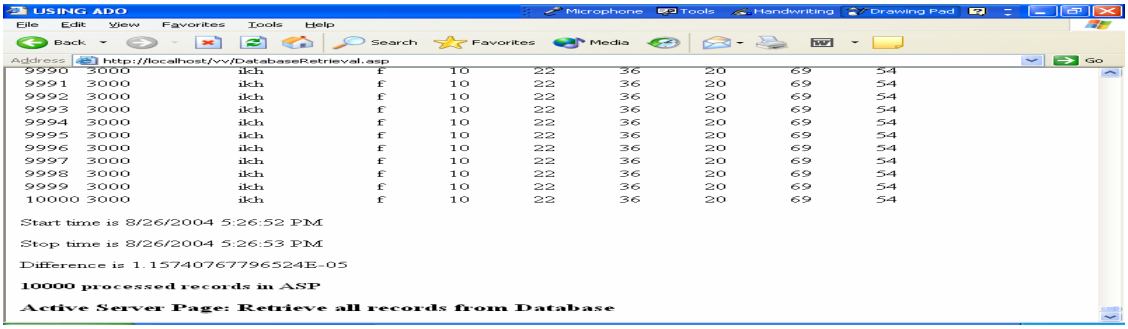

Figure 6-4. Database retrieval output using Active Server Pages.

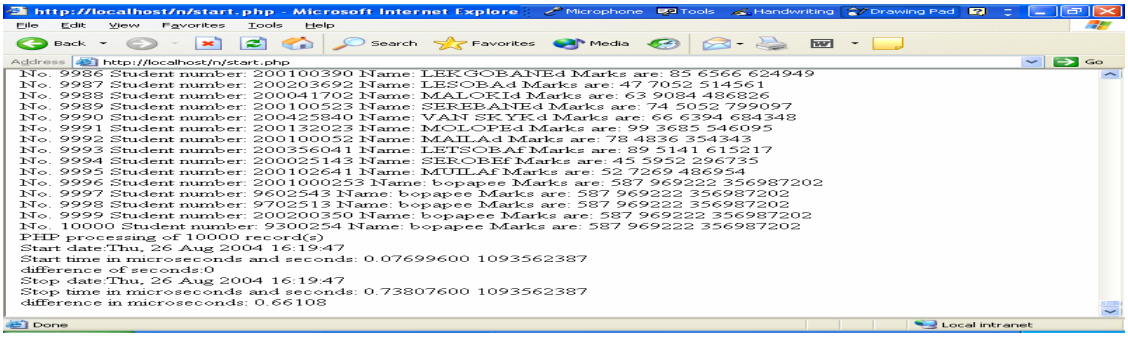

Figure 6-5. Database retrieval output using Personal Home Page (PHP).

6.4.2 Latency with scaling result

The estimated average time taken for the database retrieval in all the platforms is presented in tabular form as given in table 6.3 below. Avritzer and Weyuker (2004) indicate that the usual approach to ensuring scalability involves extensive performance testing of the target architecture with realistic workloads. Therefore, to estimate scalability as well, the records are doubled from 10000 to 20000 to 40000 and finally to 80000. This is simply because it is easier to duplicate the whole record than to enter new ones.

Table 6-3. Average time taken for the database retrieval in all the platforms as the number of records increases.

		Servlets	JSP	ASP	PHP
	Time in seconds (10000 records)	2.0	3.0	1.15	0.66
	Time in seconds (20000 records)	5.0	6.0	2.31	2.00
	Time in seconds (40000 records)	9.0	15.0	4.69	5.00
	Time in seconds (80000 records)	20.0			10.5
	Average time in seconds (10000 to 40000 records)	**5.33**	**8.0**	**2.71**	**2.55**
	Allocated Rank	**3rd**	**4th**	**2nd**	**1st**
	Allocated Value x Weight	**2x2**	**1x2**	**3x2**	**4 x2**
	Assigned Weight	**4**	**2**	**6**	**8**

It is significant to note that ASP and JSP are unable to cope with 80000 records using the configuration for this experiment. Though this can be solved by increasing the Random Access Memory (RAM) size, the fact is still worth noting given that all programs are

executed on the same computer and that most students may not be able to afford larger RAM.

From table 6.3 above it is clear that for the minimum 10000 records implemented, PHP had the best performance followed by ASP and then Java Servlets. JSP had the worst performance. A pictorial view of this result is given in the figure 6.6 below.

This result is in agreement with Cecchet *et al.* (2003) who explain PHP's better performance as due to the fact that it executes as a module in the Web server, sharing the same process (address space), thereby minimizing communication overhead between the Web server and the scripts. This is unlike Java Servlets which runs in a JVM as a separate process from the Web server.

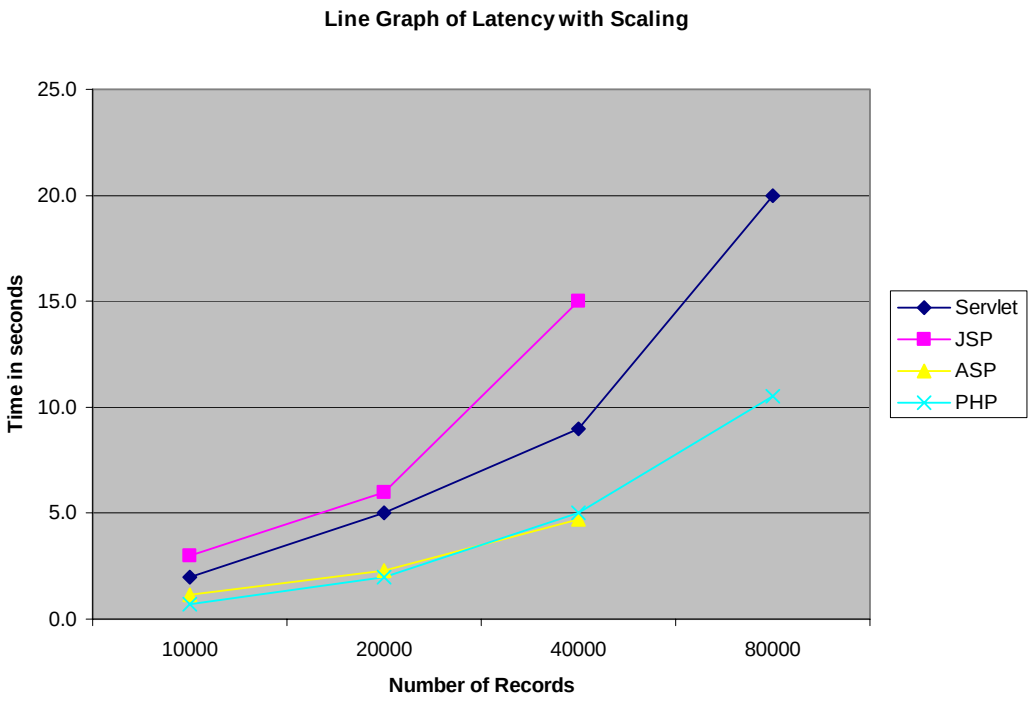

Figure 6-6. Line graph for the Latency results with Scaling.

JSP's higher latency than that of Java Servlet could be explained as being due to the fact that JSP still has to be converted to Servlet before being executed. This explanation could

also be responsible for the inability of the JSP to process 80000 records, as the compilation is already using part of the available memory and disk resources. Even though ASP is not converting to other temporary forms, as in JSP's conversion to Servlet, the interpretation of the code in the same way as for JSP also takes memory space, and so could also be responsible for the inability of the ASP to process 80000 records in this computer configuration.

For the maximum of 40000 records executable by all the platforms using the resources of this experiment's configuration, ASP seems to outperform PHP. This could be due to the mode of fetching the results. While ASP processes whole results before display, PHP displays them as it retrieves them sequentially. This would incur some fetch overhead as the number of records increases.

6.4.3 Latency with Scaling result for the MVC model of JSP and Servlet

In an attempt to see the impact of combining Java Servlet with JSP, we implemented the Model View Controller using JSP and Java Servlet. The JSP served as the HTML view fetching records via Java beans. The results for 10000 records to 80000 records are similar to those of JSP.

6.4.4 Summary

This section presents the results of the latency experiment that measured the time for the execution of programs using the selected platforms. The essence was to determine platforms that would shorten the compile-test-debug-recompile cycle in software development, especially in a practical academic laboratory. Full details of this experiment are presented in Dehinbo (2005).

In summary, a client-server-database system was set up for each of the platforms, namely Java Servlet, ASP, JSP and PHP. Programs were written to retrieve all records from a database stored on the server. The time taken from the initiation of the query from the

browser, to the time the query result was displayed on the client browser were measured for each of the four platforms as an estimate of their performance.

We combined the latency experiment with scalability by estimating the performance on 10000 records, 20000, 40000 and 80000 records. The results showed that PHP had the best performance for 10000 records, while ASP tended to outshine PHP for 40000 records. It is evident that, on average, PHP has the best average performance or latency, followed by ASP, JSP and lastly Java Servlets. PHP's result is in line with other studies such as Cecchet (2003).

Fast execution of Web applications will increase productivity in the practical laboratory. It will reduce the time spent in testing the developed applications, as well as the quality of the developed applications through extensive testing. This is based on the premise that since there is a need for recurrent testing in a practical session, a platform that has minimum latency for executing developed applications will be more desirable.

6.5 Results of the evaluation on the cost of the platforms

Table 6.4 below reflects the actual scoring for the platforms based on the cost of acquiring the platform and its associated components. A review of various literature and from programming experience led to a summary of the results on the cost criteria for the four dynamic platforms evaluated which is provided below.

Platforms that are available or downloadable free of charge give students the opportunity of obtaining their own copies with which they can practice extensively (McDougal & Boyle, 2004). The Java Servlet and the JSP engine are available free of charge at www.sun.com (Sun Microsystems, 1997). ASP, however, comes with the purchase of a Microsoft operating system only. PHP is free and open-source, so students can exploit PHP's technical strengths and lower their operating costs at the same time (Bakken *et al.*, 2002a). PHP is available for free download in source form and binaries for several platforms at http://www.php.net/ (Yager, 2001).

Table 6-4. Scoring for the platforms based on the cost criteria.

	Criteria questions	Servlet	JSP	ASP	PHP
1	Is the platform available or downloadable free of charge?	3	3	2	3
2	Is the associated Web server available or downloadable free of charge?	3	3	2	3
3	Is the related software such as the DBMS and its driver programs available or downloadable free of charge?	3	3	2	3
4	Are the Web server and the DBMS available together with the platform as a downloadable package, thereby enhancing their compatibility?	1	1	2	3
5	Do the Web server and the DBMS together with the platform come free along with the purchased operating system or other software?	2	2	3	2
	TOTAL	12	12	10	14

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

Like the Java Servlet and the JSP engine, the tomcat Web server is available free of charge at www.jakarta.com (Apache, 1997). ASP's Personal Web Server (PWS) or the Internet Information Service (IIS) comes with the purchase of a Microsoft operating system. PHP can be used on most Web servers today (Bakken *et al.*, 2002a). It can use the Tomcat Web server, which is free, and also the PWS or the IIS that comes with Windows.

Similarly, there are many free DBMSs that work with the Java Servlet and the JSP engine although they can also use Microsoft Access. ASP uses mainly DBMS like Microsoft Access and SQL Server which are not free. While PHP supports a wide range of databases (Bakken *et al.*, 2002a) and can also use Microsoft Access, unlike others it sometimes comes bundled with MySQL which is free. Being bundled together also increases the degree of interoperability between PHP and MySQL, as they are automatically installed and configured (Bakken *et al.*, 2002a).

But even if the platform and the DBMS do not come bundled together, being developed by the same organization increases the degree of interoperability. This is the situation between ASP and Microsoft Access. Moreover, the fact that a greater proportion of people (in our immediate environment) are already using Microsoft, it is as if the Web

server and the DBMS for ASP are free, being “shipped” with Microsoft Windows and Microsoft Office respectively.

6.6 Results of the evaluation on the portability of the platforms

Table 6.5 below presents the scoring for the platforms based on the criteria on portability, architectural neutrality, generality and familiarity. A summary of the results for the four platforms evaluated, based on the review of various literature and programming experience is presented below.

Table 6-5. Scoring for the platforms based on the criteria on portability, architectural neutrality, generality and familiarity.

	Criteria questions	Servlet	JSP	ASP	PHP
1	Is the platform architecture-neutral by allowing implementation of defined features, e.g. do the target processor, operating system and the compiler determine the storage allocated for primitive data types?	3	3	1	1
2	Does the code run on all implementations of the platform, thereby increasing portability?	3	3	1	3
3	Can the code also execute on platforms installed in both Intel-based and Macintosh?	3	3	1	3
4	Is the platform restricted to using a specified Web server?	3	3	1	3
5	Is the platform restricted to using specified DBMS?	3	3	2	3
6	Is the platform proprietary to a specific operating system?	3	3	1	3
7	Is the platform backward compatible?	3	3	2	3
8	Does the platform support interconnection and have facilities for talking to other services?	3	3	2	3
	TOTAL	24	24	11	22

Scale: 3 = "Yes", 2 = "Not quite or with some workaround", and 1= "No".

It is important that the platform be architecture-neutral by disallowing implementation defined features. The Java platform provides architectural neutrality in part by allowing no implementation-defined features. For example, the target processor and the compiler does not determine the storage allocated for primitive types, but the storage mapping for primitive types is defined for all implementations on all target processors (Sebesta, 1996, p.19).

Moreover, the data types in the platform should be independent of the underlying hardware. According to Hamilton (1996) and Van Hoff (1997), Java data types are independent of the underlying hardware and operating system. For example, an *int* data type is always 32 bits, regardless of whether the Java code is executing on a 32-bit or a 64-bit CPU.

ASP is not architecturally neutral because it is primarily designed to run on Windows. A search on the Internet fails to bring up any information on the use of ASP on other operating systems, or on Macintosh.

PHP is not architecturally neutral because it allows implementation-defined features. The target processor and the compiler determine the storage allocated for primitive types as in C and C++. For example, Lerdorf and Tatroe (2002, p.24) indicate that the range of acceptable values for integers varies according to the details of the platform, but typically extends from –2,147,483,648 to +2,147,483,647.

The use of the platform on a wide range of operating systems is an enhancement of portability. PHP can be used on all major operating systems, including Microsoft Windows, Linux, many Unix variants (such as HP-UX, Solaris and OpenBSD), Mac OS X, and probably others (Bakken *et al.*, 2002a). Apart from ASP, all the other platforms are well established on the Unix/Linux platforms. PHP assumes that most serious implementations are done on Unix systems while development is usually completed on Windows, and cross-platform development has thus been a major goal in the development of PHP (Lerdorf & Tatroe, 2002, p.359). PHP strives to ensure that program codes designed on Windows are portable to Linux without any problems.

Though Intel-based computer systems are more common in South Africa, it is nevertheless useful for the sake of internationalization to consider the possibility that code could also execute on platforms installed on Macintosh. The Java-based platforms as well as the tomcat Web server work on the Mac OS X (Sun Educational Services, 2000). ASP is proprietary to Windows. PHP can be used both on Intel-based

Microcomputers as well as on Macintosh computers using the Mac OS X (Bakken *et al.*, 2002a), though it may require some installation variations and some modifications to the code to port it to other implementations.

A crucial factor in choosing a Web platform is the choice of Web server. It is advantageous if the platform is not restricted to using a specific Web server. Java Servlets and JSP work on a variety of Servers. ASP, however, typically uses IIS or the PWS, both developed by Microsoft. PHP supports a large number of Web servers including Apache, IIS, PWS, Netscape and iPlanet servers (Bakken *et al.*, 2002a).

Another "plus" to portability is not being restricted to using specified DBMS. The Java-based platforms support various database systems. ASP supports only a limited number of database systems, however, being proprietary mainly to Microsoft products. However, according to Bakken *et al.* (2002a), one of the strongest and most significant features in PHP is its support for a wide range of databases.

In order to port existing programs to new systems, it is important that the platform be backward compatible, so that a script written for an older version can (ideally) run without changes in a newer version. The Java platforms are backward compatible. A command that is no longer useful is deprecated over time. In ASP, however, it is a common belief that ASP 3.0 is not fully "covered" in ASP.NET (Fuecks, 2004). Bakken *et al.* (2002a) feel that, for the most part, the developers of the PHP language have tried to be backward compatible, though in practice some changes will usually have to be made to the old program in the new implementation.

Moreover, for portability, it is necessary that the platform supports interconnection and has facilities for "talking" to other services. The Java-based platforms interoperate with other technologies via the three popular models: HTTP/CGI, RMI and CORBA/IIOP. Okuthe and Bishop (1999) found that, in addition to efficiency and flexibility, CORBA/IIOP results in greater system interoperability, which is a basic requirement of the Internet.

ASP is rated low as far as interoperability is concerned, due to the fact that it is Windows specific and based largely on the Microsoft IIS Web server (Yank, 2001). PHP has support for talking to other services using protocols such as NNTP, POP3, HTTP, COM (on Windows) and countless others (Bakken *et al.*, 2002a). PHP also has support for the WDDX complex data exchange between virtually all Web programming languages. And for interconnection, PHP has support for instantiation of Java objects and using them transparently as PHP objects. In addition, one can use the CORBA extension to access remote objects (Bakken *et al.*, 2002a).

Java Servlet, JSP and PHP use familiar syntax peculiar to C/C++. Okuthe and Bishop (1999) report that the major advantage of Java with CORBA/IIOP stems from the fact that they permit client and server objects written in different languages, running on different operating systems and on disparate networks, to collaborate. However, ASP's syntax is different from the C-style syntax shared by C/C++, Java, Perl, PHP, *etc*. (Yank, 2001), making it difficult to port programs to these other platforms. But ASP does allow one to combine VBScript and Javascript with an expandable set of software components that are normally used by C/C++ developers (Yank, 2001).

Furthermore, according to Yank (2001), PHP is fairly easy to learn especially if one has C/C++, Java or Pear experience. Lim (2004) admits that even VBScript developers with some JavaScript experience are already halfway up the learning curve, as PHP looks very like Javascript with "$" signs in front of variables. Moreover, even existing ASP programs can be ported easily to PHP with the use of "ASP2PHP", an automated VB to PHP converter (Fuecks, 2004; Lim, 2004).

6.7 Summary

This chapter evaluates the platforms according to the criteria that ensure the ease of use, speed of use (performance or latency), affordability and the portability of the platforms. It

is of particular importance that the platforms are easy and fast to use in pressing circumstances such as in a laboratory practice session or in an examination.

Affordability implies that the platform should be available at a reasonable price, so that students can own it to be able to practice what is being taught at school in their own time. Portability will also ensure the continuation of their work on major operating systems, even in their future careers, thereby guaranteeing the usefulness of their knowledge in the immediate work environment.

Table 6-6. Total scoring for the platforms based on the criteria on ease of use, latency, cost and portability.

	Criteria categories	Servlets	JSP	ASP	PHP
	Availability of smart Integrated Development Environment (IDE)	64	63	58	58
	Features to simplify execution and increase latency	37	38	45	47
	Assigned Latency score	4	2	6	8
	Cost of the platform and related software	12	12	10	14
	Architectural neutrality and portability of the platform	24	24	11	22
	TOTAL SCORE	141	139	130	149

The summary of the results is presented in table 6.6 above. Scores were given to the platforms using the criteria based on features of smart IDEs, features to simplify execution, cost of the platform and related software, as well as architectural neutrality and portability. PHP has the highest scores. This could be attributable to the following reasons:

- ✓ The availability of smart IDEs such as PhpEd and EasyPHP, which further simplify availability and installation by including the Apache Web server, PHP and MySQL as a downloadable package.

- ✓ Good performance, being faster than VBScript and Jscript (Lim, 2004), as also confirmed by our experiment.

- ✓ The free and open source nature of PHP, the IDEs, the MySQL DBMS, and even some operating system (such as Linux) on which it works. Yank (2001) states that this is a boom for developers on budget because they can set up a complete Linux-based Web server with PHP support and not pay a cent for software!.

- ✓ The cross-platform nature that allows PHP to integrate successfully with all major Web servers on all major operating systems (Yank, 2001). The portability is further enhanced by the similarity of its syntax to C/C++, Java, Perl etc., an automatic converter of ASP programs (Fuecks, 2004; Lim, 2004), as well as its support for talking to other services (Bakken *et al.*, 2002a).

PHP's score is followed by that of the Java-based platforms, which share the features expressed for PHP above, but suffer from the fact that they are considered difficult to learn. The Java-based platforms are considered more suited to serious developers who want the ultimate in power, flexibility and scalability and do not mind paying for it in sweat, tears and development time (Yank, 2001).

ASP's score is the lowest according to the criteria used in this chapter. Though it also has good IDE and fair performance, its drawbacks stem from being largely Microsoft based. This has the implication of being costly (as one has to buy the Microsoft components) and less able to interoperate with other platforms.

The developed framework has already been presented in Chapter 4, and tested in Chapter 5 and chapter 6. The next chapter is devoted to the conclusion, recommendations and review.

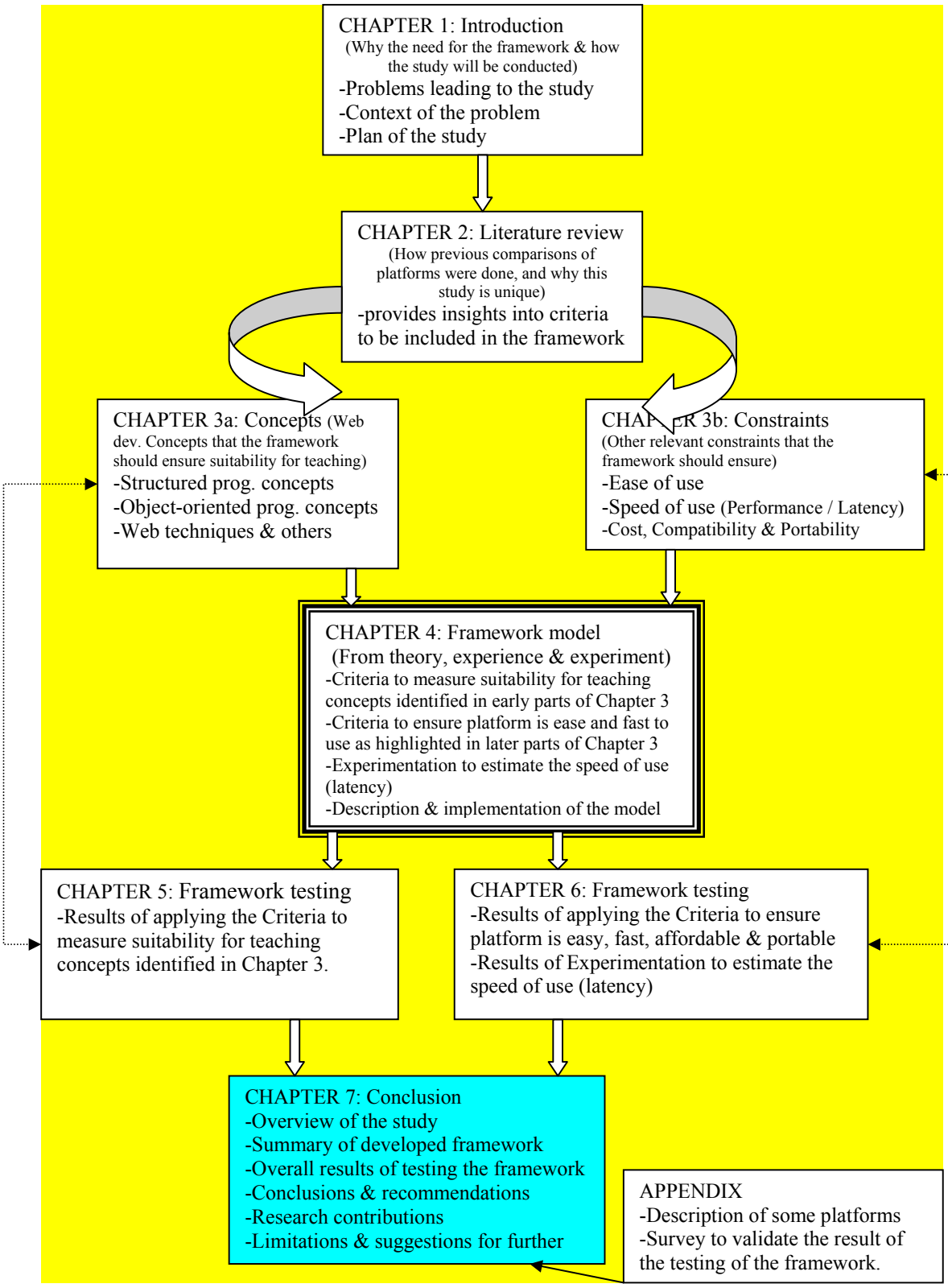

Figure 7-0. Dissertation map highlighting the relativity of chapter 7 to the rest of the dissertation.

CHAPTER 7

7 CONCLUSION, RECOMMENDATIONS AND REVIEW

7.1 Introduction

The overall goal of this study is to develop and apply a conceptual framework that can be used to determine a suitable platform for teaching dynamic Web application development as part of the curriculum for the Web Management specialization of the National Diploma in Information Technology at tertiary institutions in South Africa. The study involved the development of various criteria and experiment as part of the framework as well as the testing of the framework by using it to evaluate selected platforms.

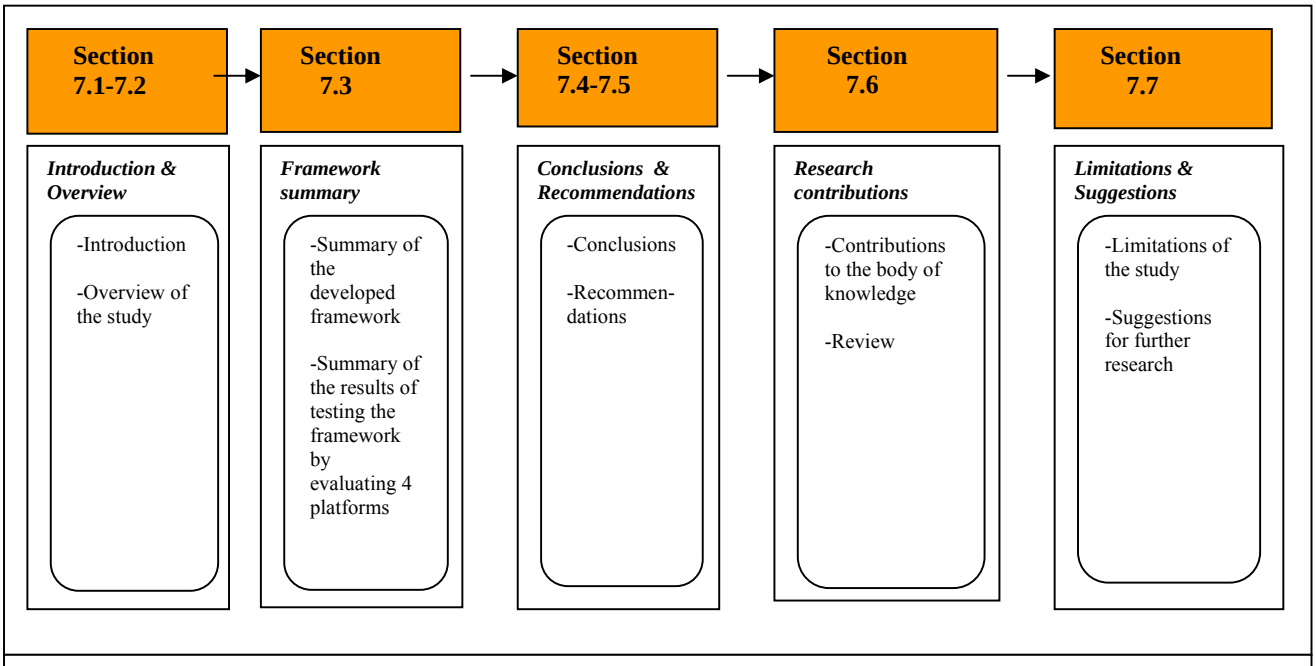

Figure 7-1. Pictorial outline of chapter 7.

This chapter concludes the study. It summarizes the framework, combines the result of testing the framework and reviews the contributions of the study.

As illustrated in figure 7.1, section 7.2 of this chapter gives an overall overview of the study. Next we present in section 7.3, a summary of the developed framework and the overall results for the testing of the framework, and a brief discussion leading into the conclusion and recommendations of the study in section 7.4 and section 7.5 respectively.

The research contributions are given in section 7.6 along with an assessment of the reliability and contributions of vital components of the research effort. Answers to some questions are provided to assess the contributions that the study makes to the body of knowledge. This includes a discussion on how each chapter has contributed towards addressing the research questions and ensures the logical flow towards achieving the set goal. Lastly, in section 7.7 we present the limitations of the study and suggestions for further research.

7.2 Overview of the study

Chapter 1 examined the context of the problems leading to the study and justified the need for the research. The major problem stemmed from the proliferation of programming and scripting languages as well as Web development platforms and the need to determine the most suitable one to enhance comprehension and utilization by students. This necessitates a scientific framework for selecting a suitable platform. Chapter 1 then presents the research questions and the research objectives. Chapter 1 also contains the blueprint for the approaches to the research methodology adopted in the study.

This is followed by the literature review in chapter 2. This reviews approaches to the analysis and comparisons of programming languages and dynamic Web platforms. From the literature review, it was argued that in comparing dynamic Web platforms, it is not sufficient simply to list the advantages and strengths of each platform. Rather, these should be examined and ranked in the light of certain desired qualities relevant to a specific “school of thought”. Certain strengths may be a priority to specific “schools of thought” and others not. For example, in training institutions, the ease of use and

suitability of a platform for teaching programming language increases our productivity more than the flexibility of the programming language.

The literature review addresses the need for the framework to be specific and yet comprehensive. Unlike other reported studies, this study attempts to solve the research problems using different approaches, thereby exploiting their strengths.

Finally, to assist in moving forward towards establishing the framework, studies that have identified some barriers to programming and pitfalls for beginners in programming are discussed. This led us to a search for studies that have designed various systems in order to solve the barriers to programming. These enabled us to identify and confirm certain criteria that can be incorporated into our framework.

Chapter 3 identifies and establishes those concepts of Web application development that should be taught as part of the curriculum for second year students at a tertiary institution specializing in Web application development. This will allow us to identify criteria for evaluating the suitability of the platforms for teaching identified concepts. Chapter 3 also identifies other constraints that the framework should satisfy. These include ease of use, ease of learning, speed of use (performance or latency), affordability and portability.

Chapter 4 presents the developed framework. It establishes the criteria to evaluate the suitability of the platforms. This involves identifying measurable qualities that will ease the teaching of the various concepts of Web application development identified in chapter 3, as well as qualities that will ensure that the platform satisfies other necessary constraints such as being easy and fast to use by the students. The criteria to ensure that the platform is fast to use are complemented with the design of the latency experiment.

The chapter also establishes the criteria for ensuring that the platform is portable and affordable, enhancing individual learning by students. Lastly, the chapter presents a description of the framework and the implementation of a spreadsheet tool for applying

the framework thereby automating the calculations and allowing variations of the values and weights of the criteria.

Chapter 5 presents the results of the testing of the framework with specific reference to the requirements of the curriculum at the Tshwane University of Technology (TUT). This involves the evaluation of four selected platforms according to the criteria set to evaluate their suitability for teaching specified concepts of Web application development.

Chapter 6 presents the results of testing the framework using the criteria set to evaluate the ease of use, speed of use, affordability and portability of the Web application development platforms. This evaluation uses weights for the various criteria as suited to the TUT requirements. The first section looks at the results based on the criteria to ensure the availability of desirable features of smart Integrated Development Environments for the platform. The next section of chapter 6 considers results on features that will simplify execution and increase the latency of programs.

This is followed by the results of the latency experiment to estimate the execution speed for programs written using each of the dynamic Web platforms. Lastly, it contains a section that presents the results on the affordability, architectural neutrality and portability of the platforms.

Finally, chapter 7 concludes the study. Here, an overview of the study is presented. We summarize the overall data collection approach and the developed framework. This is followed by a combination of the results of the testing of the framework, leading to the conclusion and recommendations. The last part reviews the conclusion and recommendations in order to ensure confidence in their integrity and validity. The research contributions and their implications are discussed and evaluated. The limitations of the research and potential areas for further research are also discussed.

Appendix 1 discusses the different dynamic Web platforms that could be used for teaching Web application development, and the reasons leading to the selection of four of

the platforms for testing the framework. Appendix 2 contains an informal survey that was used to document the experiences of past students on the four selected platforms, and to determine whether these experiences correspond with the results of the testing of the framework.

7.3 Summary of the developed framework and its testing

Table 7.1 below presents the developed framework as well as the results of testing the framework. The first component is the criteria to evaluate the suitability of platforms for teaching the basic concepts of programming as well as other Web application development concepts. This is followed by the criteria established to evaluate the ease of use, speed of use, as well as affordability and portability of the platform. Lastly, the latency experiment component measures the speed of execution of Web programs. The survey is simply included for informal documentation of past students' experiences on various Web platforms.

The testing of the developed framework was conducted using four selected platforms. The results given in chapter 5 and chapter 6 are combined in the overall results given in table 7.1 below. The purpose of combining the results is to determine the platform with the overall best score.

From the results in table 7.1 below, it is evident that PHP is considered most suitable in each result category namely;

- ☑ the category for the suitability for teaching Web development concepts,
- ☑ the category for the ease of use, latency, affordability and portability,
- ☑ the category for the latency experiment.

Java Servlet and JavaServer Pages are considered least suitable based on all the criteria, though, when the latency result is added, JSP becomes the most unsuitable. However, Java Servlet is ranked equal to PHP in suitability for teaching object-oriented programming concepts.

Table 7-1. Summary of the developed framework and the overall result of its testing.

	Criteria categories	Servlets	JSP	ASP	PHP
	Basic concepts				
	Lexical Structures	19	21	28	24
	Data types	13	13	22	27
	Expressions and Operators	18	18	25	26
	Control Structures	11	11	23	19
	Functions and modularizations	13	11	9	12
	TOTAL SCORE	**74**	**74**	**107**	**108**
	Other Web development concepts				
	Object-oriented programming	33	33	21	33
	Web techniques	29	29	33	31
	Database handling capabilities	11	11	15	16
	File processing capabilities	25	25	25	30
	Exception handling capabilities	19	19	19	24
	XML support	15	15	15	15
	TOTAL SCORE	**132**	**132**	**128**	**149**
1	**Overall suitability score for teaching Web dev. concepts**	**206**	**206**	**235**	**257**
	Ease of use, latency, affordability and portability				
	Availability of smart Integrated Development Environment (IDE)	64	63	58	58
	Features to simplify execution and increase latency	37	38	45	47
	Cost of the platform and related software	12	12	10	14
	Architectural neutrality and portability of the platform	24	24	11	22
2	**TOTAL SCORE**	**137**	**137**	**124**	**141**
3	**Assigned Latency score**	**4**	**2**	**6**	**8**
	OVERALL SCORE	**347**	**345**	**365**	**406**

Taking a closer look at Java Servlet and JSP, JSP had the least score on the whole, although in some cases it ranked the same as Java Servlets while in other cases it ranked the same as ASP. This is to be expected considering the fact that JSP shares the same background language with Java Servlet, and also shares logic similar to ASP's way of embedding code within HTML tags. This suggests that while JSP gained from the

simplicity of its structure by simply embedding code in HTML, this same simplicity requires later translation into Servlet thereby obstructing high latency.

JSP ranked second along with Java Servlet in the category of the *ease of use*, latency, affordability and portability. This could be explained as due to the background power of the Java Programming language, the fact that it is free, its matured and free IDEs as well as the interoperability of the Java environment with other technologies such as various DBMS, middleware technologies such as RMI, CORBA and the recent mobile technologies.

While ASP ranked second most suitable on the whole, it ranked lowest in the category of suitability for teaching object-oriented programming concepts. This is an area that Windows.NET aims to address. ASP's ranking higher than JSP could be explained by the familiarity with Microsoft's look and feel in many related products.

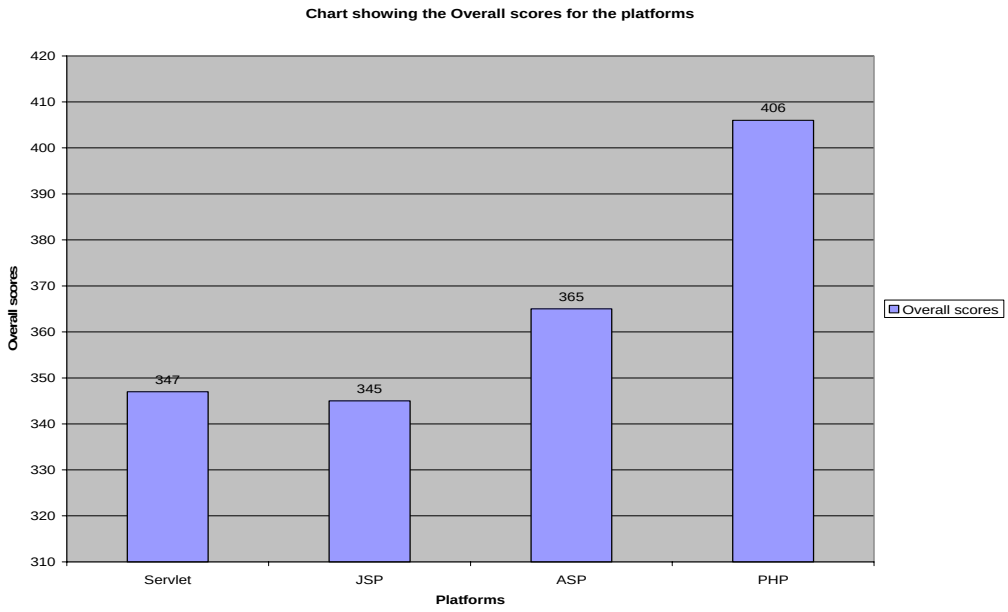

Figure 7-2. Chart showing the total scores for all the platforms.

Finally, it is worthy to note that the ranking of the overall results, as shown in figure 7.2 above, differs slightly from the trend in the rankings in the results of the survey in the

Appendix 2. The difference is that the survey results for JSP are higher than those for Java Servlets. This is because students are interested in the simplicity of structure in JSP and have little regard for other important criteria. This highlights the relevance of the study as it considers results using various criteria and looks at the problem from different angles.

7.4 Conclusion

In this study we developed and applied a conceptual framework that can be used to determine a suitable platform for teaching dynamic Web applications development as part of the curriculum for the Web application development specialization of the National Diploma in Information Technology at tertiary institutions in South Africa. The framework has taken into consideration the need to satisfy certain requirements that are considered important to the effective comprehension and future usefulness of Web application development concepts. The requirements, later refined into criteria for the framework is aimed at ensuring:

1. Effective teaching of dynamic Web applications development: This is achieved by incorporating criteria that will accomplish the following:

- Ensure suitability for the purpose of teaching Web applications development for second year students of tertiary institutions, by building upon the program design concepts taught in the first year of study.
- Ensure suitability for real life programming that students will be exposed to as soon as they graduate and also offer integration with other relevant components of the curriculum of the of Web applications development specialization of the National Diploma in Information Technology.
- Ensure affordability to allow for individual ownership of the chosen platform by students, so that they can practice on their own.
- Enhance comprehension by using commands that have a reasonable number of unique basic syntactical components with syntax, semantics and meanings that make commands easy to remember. This avoids redundancy in some type names

and commands. Black (2004) indicates that such redundancy clutters program code without adding any extra information.

2. Allowing an efficient teaching environment: This is achieved by incorporating criteria that allow for an efficient teaching environment that does not distract students from their purpose, by accomplishing the following:

- Ensuring suitability for completing tasks within a reasonable time period with features that promote good readability of code, tracing and debugging aids with useful and smart integrated editing environments equipped with context sensitive helps.
- Ensuring high throughput by compiling and executing quickly to ensure suitability for the recurrent testing and debugging in a practical laboratory.
- Further ensuring affordability through the requirements for the use of minimum disk space and memory resources, with simpler architecture to facilitate speedy operations, and simpler execution steps because of higher-level commands.
- Ensuring portability among major operating systems through the affordability, architectural neutrality, generality and familiarity of the operating environment.

The criteria for the framework, refined from the requirements using document analysis includes the suitability for teaching Web application development concepts, ease of use for second year students as well as speed of use. The speed of use is complemented with a latency experiment to estimate the speed of execution of programs written using the selected platforms. Other criteria are aimed at ensuring affordability of the platform by students as well as portability of the platform across major operating systems.

The development of the framework contributes to the body of knowledge by being unique. Previous studies either simply compared platforms without an established comprehensive yet specific framework or compared them only in terms of a specific factor. Most importantly, most comparisons were not done with reference to specific utilization. Moreover, the framework is backed with useful information from previous relevant studies, established theoretical references, experiment and experience.

The purpose of the application of the framework is not to persuade readers that one dynamic Web platform is better than another, but rather to help readers make a more informed decision as to which platform is more suitable for teaching Web application development in tertiary institutions. This implies that the framework should ensure that although the chosen platform may or may not be the most powerful or most common in the market, it is the one that has features that will contribute to the accomplishment of Web application development expertise by second year students of the National Diploma in Information Technology.

The framework is tested using four selected platforms. The results of the testing indicate that of the four dynamic Web platforms evaluated, PHP is considered most suitable. This is probably due to the fact that it is suitable for teaching both structured programming and object oriented programming concepts. Moreover, it is free, portable and equipped with a free easy-to-use IDE as well as free DBMS (MySQL). It also supports various Web servers, DBMS and interoperates with other technologies.

In essence, this study will be of great value in the teaching of Web applications development. In conclusion, by being unique, the study provides an addition to the body of knowledge in the form of a framework for choosing a suitable dynamic Web platform to enhance teaching of Web application development at tertiary institutions in South Africa. This provides the possibility of increasing the learners' potential in such a way that could lead to higher productivity in the software development and services industries in the future.

7.5 Recommendations

Based on the essence of this study, it is recommended that comparisons of platforms be done with specific uses and criteria in mind. Rather than listing the advantages and strengths of each platform in a comparison, the advantages and strengths of each platform should be examined and ranked in the light of certain desired qualities relevant to specific

"schools of thought". This is because certain strengths may be of highest priority to specific "schools of thought" only. For example, at the University of Technologies, the ease of learning a programming language increases our productivity more than the flexibility of the programming language or platform.

In addition, we recommend that frameworks for the evaluation and comparisons of programming languages and Web platforms should be comprehensive, yet still providing users with the freedom to include and exclude some criteria or to vary the degree of the relative importance of various criteria.

In order to make an informed choice of a platform for teaching Web application development at tertiary institutions, we recommend the evaluation of selected platforms using our developed framework containing various criteria. The most important of such criteria is to ensure that a chosen platform is suitable for teaching identified concepts. Other criteria include ease of use, speed of use, affordability and portability. Moreover, the framework is supported with evidence from relevant theoretical literature references, information from previous relevant studies, experience and programming experiments.

7.6 Research contributions

This section focuses on the contributions of the study. The main contribution of the research is the addition to the body of knowledge in the form of a framework for choosing a suitable dynamic Web platform for teaching Web application development in tertiary institutions.

Further contributions emanate from the insights provided into the analysis of various features of specified platforms. We need no longer rely only on features promised by the platform development teams, since according to Olivier (1999, p.41), they might contain marketing or publicity hype. Neither do we rely simply on general comparisons. The choice of a platform using our developed framework provides a level of certainty that

teaching Web application development using such a platform will be of great service to our students.

Another contribution lies in the integration and application of knowledge of most of what the undergraduate Information Technology students of Web application development are required to learn during their study. These include knowledge of the Internet and the World Wide Web, HTTP, Networking, HTML, Web platforms, and Database Management Systems using SQL. Students reading the study could therefore benefit in one aspect or another. This is an evidence of how research aids teaching.

Apart from the overall research contributions of the study, questions can be asked on how we can further assess the reliability and contribution of vital components of the research effort. The following questions and answers will be of use in this regard.

How is this study unique? What distinguishes it from other related studies?

This study is not aimed only at comparing dynamic Web platforms as in other related studies. Rather, we developed a unique framework containing a set of criteria that were used to evaluate the various platforms in terms of their suitability for teaching Web application development to second year undergraduate students. Moreover, the framework caters for the satisfaction of certain constraints that are considered important to the use of the selected platform by students.

Is it likely that the study will lead to improved productivity in students?

The use of a Web development platform that is well suited for teaching students will enhance the comprehension of the Web application development concepts and reduce the period it takes to develop specific applications. Being easy and fast to use will also reduce the period it takes to implement specified actions, as well as the time and effort needed to work around a solution to a problem. Saving time and effort means saving money and increasing productivity. Being affordable and portable will furthermore enable continuation of practice sessions started in the practical laboratory.

Does the research work reflect seasoned thinking, conveying completeness and thoroughness?

The research problem, the developed criteria, as well as the experiment were viewed from different angles. Various research approaches were used thus enhancing thoroughness, and the diverse forms of the framework's testing results were presented from multiple perspectives before being combined. The last chapter of the dissertation contains a review of the research and, in particular, the contributions made by the research study. This indicates thoroughness and reflection on the part of the researcher.

Is the dissertation well written? Does it flow logically? Are the central ideas easily accessed?

At a workshop on evaluating and assessing theses and dissertations, conducted at the former TNG on the 6th of November, 2003, Prof. Chris Kapp of the University of Stellenbosch mentioned two common approaches to the writing of research theses and dissertations. These are the "Biblical" approach in which the study is presented logically from introduction to conclusion, and the "Journalistic" approach, where the main point of the study is presented first, before the rest of the report.

This study used a combination of the two approaches. The summary of the research is presented both in the abstract, in chapter 1, as well as pictorially at the beginning of each chapter. Thereafter, the report flows logically from introduction to conclusion. Within chapters, logical flow of information has also been maintained.

In order to ensure the quality of writing in this dissertation, the services of a professional language editor have been engaged. The table of contents, the abstract, dissertation map, the introductory overview and the brief insight into the research as well as the pictorial layout at the beginning of each chapter allow for easy access to the central ideas in the dissertation.

Is the topic of contemporary interest to scholars, academic readers and practitioners in this area?

The rate at which organizations are moving transactions to the Web is tremendous. Many Web applications developers will be needed to satisfy this demand. Moreover, many academic readers, even from other disciplines, are interested in implementing their own Web applications. Such people will be interested in a framework for determining a platform that is easy to learn and use, fast to use, affordable and portable.

How rigorous is the study?

In order to arrive at the conclusion, we have studied the background to the problem domain, reviewed relevant literature, used various research approaches including document analysis, programming experiment, and documented survey in the Appendix. We reviewed the use of various platforms in order to develop the desired Web application development concepts to be taught and the criteria for evaluating the platforms. Then we tested the framework by evaluating four platforms using information sought from various references, experiences, observations and developed programs using the four selected platforms.

7.7 Limitations and suggestions for further research

It is necessary to identify limitations to a study in order to pinpoint areas for further research. The scope of this study does not allow for incorporation into the framework, detailed syntactical evaluation of the commands and functions of the dynamic Web platforms. This could be the subject matter of the author's future studies.

8 Reference List

ADEYEMO, A.M. 2000. Towards a framework for the evaluation of websites. M.Com. dissertation, University of Pretoria.

ADIDA, B. 1997a. It all starts at the server. *IEEE Internet Computing*. **1** (1) 75-77.

ADIDA, B. 1997b. Database-backed websites. *IEEE Internet Computing*. **1** (6) 78-80.

APACHE. 2004. *Apache Tcl: Why Tcl?* [Online]. Available from: http://tcl.apache.org/why.rvt [Accessed: 06/02/2004].

APTE, V., HANSEN, T. & REESER, P. 2003. Performance comparison of dynamic Web platforms. *Computer Communications*. **26** (8) 888 – 898.

ASHENFELTER, J.P. 1999. *Choosing a Database for Your Web Site*. New York: John Wiley & Sons.

ATINAV Inc. 2003. *aveConnect JDBC Driver for MS Access* [Online]. Available from: http://www.atinav.com/products/aveconnect/MSAccess.htm [Accessed: 06/02/2004].

AVRITZER, A. & WEYUKER, E.J. 2004. The role of modeling in the performance testing of e-commerce applications. *IEEE Transactions on Software Engineering*. **30** (12) 1072-1083.

BAKKEN, S., AHTO, J., AULBACH, A., BECKHAM, D. & CASTAGNETTO, J. 2002a. *PHP Manual*: *Getting started* [Online]. Available from: http://www.php.net/docs.php [Accessed: 02/02/2003].

BAKKEN, S., AHTO, J., AULBACH, A., BECKHAM, D. & CASTAGNETTO, J. 2002b. *PHP Manual*: *Language reference* [Online]. Available from: http://www.php.net/docs.php [Accessed: 02/02/2003].

BERGIN, J. 1996. Object Technology in the Classroom - Java as a Better C++. *ACM SIGPLAN Curricular Patterns.* **1** (2) 21-27.

BERTINO, E. & CATANIA, B. 2001. Integrating XML and databases. *IEEE Internet Computing*. **5** (4) 84-88.

BIDDLE, R. & TEMPERO, E. 1998. Java pitfalls for Beginners. *ACM SIGCSE Bulletin inroads*. **30** (2) 48-52.

BISHOP, J. 1998. *Java Gently: Programming principles explained*. 2nd edition. London: Addison-Wesley Longman.

BISHOP, J. & HURTER, R. 1999. Competitors to Java: Scripting languages. (Paper read at the South African Computer Lecturers Association -SACLA- conference, June, Golden Gate, South Africa). Unpublished.

BLACK, A.P. 2004. Post-Javaism. *IEEE Internet Computing*. **8** (1) 93-96.

BOTHA, R. & ELOFF, J. 2002. An Access Control Architecture for XML Documents in Workflow Environments. *South African Computer Journal*. **28** (June) 3-10.

BOWLING, A. 2002. Research methods in health. 2nd edition. Buckingham: Open University Press

CECCHET, E., MARGUERITE, J. & SWANENEPOEL, W. 2002. Performance and Scalability of EJB Applications. In: *Proceedings of the 17th Annual ACM conference on object-oriented programming, systems, languages and applications.* September, 2002. New York: ACM Press.

CECCHET, E., CHANDA, A., ELNIKETY, S., MARGUERITE, J. & ZWAENEPOEL, W. 2003. Performance Comparison of Middleware Architectures for Generating Dynamic Web Content. *Lecture Notes in Computer Science: Middleware*. **2672** (2003) 242-261.

CHAPKHANOV, K. 1999. *Working with Active Server Pages: Chapter 12: Enhancing Interactivity with Cookies, Headers, and the Server Object* [Online]. Available from: http://makecashonline.tripod.com/ch12.html [Accessed: 09/09/2004].

CLARK, D. 1999. Internet technologies in the marketplace. *IEEE Internet Computing*. **3** (4) 13-15.

COCKBURN, A. & BRYANT, A. 1997. Leogo: An equal opportunity user interface for programming. *Journal of Visual Language Computing*. **8** (6) 601-619.

COERTZE, D. & HEATH, R. 1997. *Research Methodology for Technikon Students: A practical Approach.* Durban: Technikon Natal publishing.

COOPER, R. 2001. Software for managing websites. In: *Proceedings of the South African Institute of Computer Scientists and Information Technologists (SAICSIT) Annual conference*. September 2001. Pretoria, South Africa: SAICSIT.

COX, B. 2001. Web applications as Java Servlets: Just say no to JSP. *Dr. Dobb's Journal: Software tools for the professional programmer*. **26** (5) 1-7.

CRESWELL, J.W. 2003. *Research design: qualitative, quantitative and mixed methods approaches*. 2nd edition. Thousand Oaks, Calif.: SAGE.

DATTA, A., DUTTA, K., THOMAS, H., VANDERMEER, D. & RAMAMRITHAM, K. 2002. Accelerating dynamic Web content generation. *IEEE Internet Computing*. **6** (5) 27-36.

DAVIDSON, B. H. 2001. Database driven, dynamic content delivery: providing and managing access to online resources using Microsoft Access and Active Server Pages. *OCLC Systems & Services*. **17** (1) 34-42.

DEHINBO, J. 2000. Java Web Server Database Connectivity. B.Sc. Hons. project. University of South Africa. Pretoria.

DEHINBO. J. 2003. Enhancing quality of Students' Internet knowledge through in-house developed on-line testing system at TNG. In: *Proceedings of the 15th international conference on Assessing Quality in higher education, July, 2003. Cape Town* [Online]. Available from: http://www.uwc.ac.za/dll/conference [Accessed: 29/06/2004].

DEHINBO, J. 2004a. The impact of Web-based middleware systems on training and assessment through inhouse developed online testing system. In: *Proceedings of the Informing science + Technology Education (InSITE) conference*. June 2004. Rockhampton, Australia [Online]. Available from: http://proceedings.informingscience.org/InSITE2004/027dehin.pdf [Accessed: 29/06/2004].

DEHINBO, J. 2004b. Determining a suitable programming language for the B.Tech. degree. (Paper read at the South African Computer Lecturers Association -SACLA- conference, July, Durban, South Africa). Unpublished.

DEHINBO, J. 2005. The performance of Web-based 2-tier middleware systems. *Journal of Issues in Informing Science and Information Technology*. **2** (2005) 757-769 [Online]. Available from: http://2005papers.iisit.org/I59f23dehin.pdf [Accessed: 13/05/2005].

DEITEL, H.M., DEITEL, P.J. & NIETO, T.R. 2001. *e-Business & e-Commerce: how to program*. New Jersey: Prentice Hall.

DEITEL, H.M. & DEITEL, P.J. 2002. *Internet & World Wide Web: how to program*. 2nd edition. New Jersey: Prentice Hall.

DEVDAILY. 1998. *Servlet Runner 101: testing your Java Servlets with the servletrunner* [Online]. Available from: http://www.devdaily.com/java/edu/pj/pj010025/index.shtml [Accessed: 06/10/2004].

DHYANI, D., NG, W.K., & BHOWMICK, S.S. 2002. A Survey of Web Metrics. *ACM Computing surveys*. **34** (4) 469-503.

DIX, A., FINLAY, J., ABOWD, G. & BEALE, R. 1998. *Human-Computer Interaction*. 2nd edition. Hertfordshire: Prentice Hall.

DOMAIN DLX. 2003. *ASP – Setting up a DSN* [Online]. Available from: http://www.domaindlx.com/dsn_setup.asp [Accessed: 11/08/2003].

EBNER, E., SHAO, W. & TSAI, W. 2000. The five-module framework for Internet application development. *ACM Computing surveys*. **3** (1) 46-50.

EISENSTADT, M. 1983. A user-friendly software environment for the novice programmer. *Communications of the ACM*. **26** (12) 1058-1064.

ERVIN, J. R. 2000. Dynamic delivery of information via the World Wide Web. *Library High Tech*. **18** (1) 55-60.

FUECKS, H. 2004. *Is PHP the natural upgrade path for ASP 3.0/VB 6?* [Online]. Available from: http://www.sitepoint.com/blog-post-view.php?id=175686 [Accessed: 20/10/2004].

FARRELL, J. 2003. *Java Programming.* 2nd edition. Boston, Mass.: Thomson Learning.

FRANTZ, R. 1999. Try CGI, *Byte*, **03605280** (8).

FRATERNALI, P. 1999. Tools and approaches for developing data-intensive Web applications. *ACM Computing surveys*. **31** (3) 227-236.

GARSIDE, R. & MARIANI, J. 2003. *Java: first contact*. 2nd edition. Toronto: Thomson Learning

GLINERT, E. & TANIMOTO, S. 1984. Pict: an interactive graphical programming environment. *Computer*. **17** (11) 7-25.

GORTON, I. & LIU, A. 2003. Evaluating the performance of EJB components. *IEEE Internet Computing*. **7** (3) 18-23.

HADJERROUIT, S. 1998. Java as first programming language: a critical evaluation. *ACM SIGCSE Bulletin inroads*. **30** (2) 43-47.

HALL, M. 1999. *Tutorial on Servlet and JSP* [Online]. Available from: http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/ [Accessed: 15/04/2004].

HAMILTON, M.A. 1996. Java and the shift to Net-Centric computing. *IEEE Computing Practices*. **1** (2) 31-39.

HARTMAN, H. 2001. Tools for dynamic websites: ASP vs PHP vs ASP.NET, *Seybold Report Analysing Publishing Technologies*, **15339211** (12).

HOLT, R. & CORDY, J. 1988. The turing programming language. *Communications of the ACM*. **31** (12) 1410-1423.

HOLT, R., WORTMAN, D., BARNARD, D. & CORDY, J.R. 1977. SP/k: a system for teaching computer programming. *Communications of the ACM*. **20** (5) 301-309.

HORSTMANN, C. 2002. *Big Java*. New York: John Wiley & Sons.

JBUILDER. 2004. *JBuilder 9 Personal Tutorial*. [Online]. Available from: http://www.cs.dal/~arc/teaching/CS2110/TutorialS/introToJBuilderTutorial.htm [Accessed: 25/10/2004].

JCREATOR. 2004. *Using JCreator: IDE tutorial* [Online]. Available from: http://www.cs.usm.maine.edu/~welty/cos160JCreatorOnlineTutorial/ [Accessed: 20/10/2004].

jGURU, 2000. *JavaServer Pages Fundamentals, Short Course Contents* [Online]. Available from: http://developer.java.sun.com/developer/onlineTraining/JSPIntro/contents.html [Accessed: 28/05/2002].

KAPP, C. 2003. Evaluating and assessing postgraduate thesis and dissertations. *Technikon Northern Gauteng Postgraduate workshop series*. Pretoria: TNG Press.

KARPINSKI, R. 1998. Dynamic Java Pages, MS-style, *Internet Week*, **10969969** (721).

KAY, A. 1993. The early history of smalltalk. *ACM SIGPLAN Notices*. **28** (3) 69-96.

KELLEHER, C., & PAUSCH, R. 2005. Lowering the barriers to programming: A taxonomy pf programming environments and languages for novice programmers. *ACM Computing surveys*. **37** (2) 83-137.

KHAN, F. 2003. *File uploading with ASP* [Online]. Available from: http://www.stardeveloper.com/articles/display.html?article=2001042501&page=4 [Accessed: 18/10/2004].

KLOPPER, S. 2003. Comparing the three scripting languages: PHP, ASP and JSP with each other, in order to use the best option for a specific application. *Technologiae.* **1** (2003) 17-22.

KOLLING, M., QUIG, B., PATTERSON, A. & ROSENBERG, J. 2003. The BlueJ system and its pedagogy. *Journal of Computer Science Education*. **12** (4) 249-268.

KOLLING, M. & ROSENBERG, A. 1996. Blue: A language for teaching object-oriented programming. In: *Proceedings of the 27th SIGCSE Technical Symposium on Computer Science Education, Philadelphia*. New York: SIGCSE: 190-194.

KRUSE, W. 2003. A comparison of PHP and J2EE. *Technologiae*. **1** (2003) 110-117.

KURNIAWAN, B. 2001. *Uploading Files with Beans* [Online]. Available from: http://www.onjava.com/lpt/a/745 [Accessed: 18/10/2004].

KURTZ, T. 1981. *History of Programming languages*. New Hampshire: Academic Press.

LAHTINEN, E., ALA-MUTKA, K. & JARVINEN, H. 2005. A study of the difficulties of novice programmers. In: *Proceedings of the 10th annual SIGCSE conference on innovation and technology in computer science education, Caparica*, Portugal: SIGCSE: 14-18.

LEEDY, P.D. & ORMROD, J.E. 2001. *Practical research: planning and design*. 7th edition. New York: Merrill Prentice Hall.

LERDORF, R. & TATROE, K. 2002. *Programming PHP*. Sebastopol, Calif.: O'Reilly & Associates Inc.

LIM, B.L. 2002. Teaching Web development technologies: Past, present, and (near) future. *Journal of Information Systems Education*. **13** (2). pp. 117-123.

LIM, J. 2004. *PHP Everywhere Part 1: PHP, Javascript and VBScript Language Summary* [Online]. Available from: http://phplens.com/phpeverywhere/node/view/30 [Accessed: 20/10/2004].

MACKINNON, M. & DENNE, B. 2001. *The Usborne guide to the Internet*. New York: Usborne.

MARSHAK, M. & LEVY, H. 2003. Evaluating Web user perceived latency using server side measurements. *Computer Communications*. **26** (8) 872-887.

MAXIMILIEN, E.M. & SINGH, M.P. 2004. A framework and ontology for dynamic Web services selection. *IEEE Internet Computing*. **8** (5) 84-93.

MCDOUGALL, A. & BOYLE, M. 2004. Student strategies for learning computer programming: Implications for pedagogy in informatics. *Education and Information Technologies*. **9** (2) 109-116.

MCMILLAN, J.H. & SCHUMACHER, S. 2001. *Research in education*. 5th edition. New York: Addison Wesley Longman.

MENASCE, D.A. 2002. Load testing of websites. *IEEE Internet Computing*. **6** (4) 70-74.

MENASCE, D.A. 2003. Web server software architectures. *IEEE Internet Computing*. **7** (6) 78-81.

MERCER, D. 2003. *Programming the Web Using ASP.NET*. New York: McGraw Hill.

MILLER, P., PANE, J., METER, G. & VORTHMANN, S. 1994. Evolution of novice programming environments: The Structure Editors of Carnegie Mellon University. *Interactive Learning Environment*. **4** (2) 140-158.

MILNER, I. & ROWE, G. 2002. Difficulties in learning and teaching programming: views of students and tutors. *Education and Information Technologies*. **7** (1) 55-66.

MOTH, J. & EPSTEIN, D. 1998. *JJ: a language designed for beginners (less is more)* [Online]. Available from: http://www.publicstaticvoidmain.com/JJ_A_Language_Designed_For_Begin-ners_LessIsMore.pdf [Accessed: 21/01/2006].

MSDN. 2000. Introduction to Web Development Technologies. *Course number 1912 Workbook.* Part Number: X05-91011.

MSDN. 2004. Introducing Visual InterDev. *Microsoft Visual Studio Help file*. Accessed on 20 October, 2004.

OKUTHE, J & BISHOP, J. 1999. Performance comparison of object models for Web based Distribution. In: *Companion to the Proceedings of OOPLSA*. November, 1999. Denver, USA: 102-104.

OLIVIER, M.S. 1999. *Information Technology Research: A Practical Guide*, 2nd edition. Pretoria: Van Schaik.

OVERMARS, M. 2000. *Drape: Drawing programming environment* [Online]. Available from: http://www.cs.uu.nl/people/markov/kids [Accessed: 21/01/2006].

PRECHELT, L. 2000. An empirical comparison of seven programming languages. *Computer*. **33** (10) 23-29.

PRESS, W. 2003. *Object Oriented Design Principles in Visual Basic* [Online]. Available from: http://www.joelonsoftware.com/printerFriendly/articles/APIWar.html [Accessed: 20/10/2004].

PUCK PRODUCTS. 2004. *ColdFusion MX* [Online]. Available from: http://www.puck.co.uk/Products/macromedia/coldfusion%2mx.htm [Accessed: 05/02/2004].

PYTHON SOFTWARE FOUNDATION. 2003. *Beginner's guide to Python* [Online]. Available from: http://www.python.org/topics/learn/ [Accessed: 05/02/2004].

PYTHON SOFTWARE FOUNDATION. 2004. *Why work on Python* [Online]. Available from: http://www.python.org/dev/why.html [Accessed: 05/02/2004].

RENAUD, K., LO, J., BISHOP, J., VAN ZYL, P & WORRALL, B. 1999. Algon: A framework for supporting comparison of distributed algorithm performance. In: *Proceedings of PNDP conference, February 2003. Genoa*. Rome: PNDP.

ROSS, D., ZYMARIS, A. & CON, D. 2000. DB Forms, PHP, MYSQL, and PHPLIB: A reusable database framework for writing Web apps. *Dr. Dobb's Journal: Software tools for the professional programmer*. **25** (8) 11-16.

SCHNEIDER, G. & PERRY, J. 1999. *The Internet*. Boston, Mass.: Thomson Learning.

SEBESTA, R.W. 1996. *Concepts of Programming Languages*, 3rd edition. Reading, Mass.: Addison-Wesley.

SINGH, S., ERWIN, G. & KOTZE, P. 2001. Electronic Business Accepted Practices (e-BAP): standardised HCI for e-commerce in South Africa. In: *Proceedings of the South African Institute of Computer Scientists and Information Technologists (SAICSIT) Postgraduate research symposium.* Pretoria: UNISA Press: 66-73.

SINGH, S. & KOTZE, P. 2002. Towards a framework for e-commerce usability. In: Kotze, P., Venter, L. & Barrow, J. (eds.). *Proceedings of the annual research conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT) held in Port Elizabeth on 16-18 September 2002*. Pretoria: UNISA Press: 2-10.

SPOLSKY, J. 2004. *How Microsoft Lost the API War* [Online]. Available from: http://www.sitepoint.com/print/server-side-language-right [Accessed: 20/10/2003].

SUN EDUCATIONAL SERVICES. 2000. *Web Component Development With Java Technology. SL-314 Student Guide. Revision A.1*. New York: Sun Microsystems.

SUN EDUCATIONAL SERVICES. 2001. *Java Programming Language Workshop. SL-285 Student Guide. Revision A.1.* New York: Sun Microsystems.

SUN EDUCATIONAL SERVICES. 2002. *Web Component Development With Java Technology. SL-314 Student Guide. Revision A.1.* New York: Sun Microsystems.

SUN MICROSYSTEMS. 1997. *JDBC Guide: Getting Started* [Online]. Available from: http://www.pearson.com.mx/component/component_java_21_days/jdk/1.2/doc/html/guide/jdbc/getstart/introTOC.doc.html [Accessed: 11/08/2003].

SUN MICROSYSTEMS. 2001. *Java Tutorial.* [Online]. Available from: http://www.javasoft.com [Accessed: 11/08/2003].

TEITELBAUM, T. & REPS, T. 1981. The Cornell program synthesizer: A syntax-directed programming environment. *Communications of the ACM*. **24** (9) 563-573.

TELKOM INTERNET. 2003. Centres of achievement: Science and Technology Advertorial column. *Sunday World*. 2 November 2003.

TREESE, G.W. & STEWART, L.C. 2003. *Designing Systems for Internet Commerce*, 2nd edition. Reading, Mass.: Addison-Wesley.

VAN ROSSUM, G. & DRAKE, F.L. 2003. *Python Tutorial* [Online]. Available from: http://www.python.org/doc/current/tut [Accessed: 02/05/2004].

VAN HOFF, A. 1997. The case for Java as programming language. *IEEE Internet Computing*. **17** (1) 51-56.

VINOSKI, S. 2003. The performance presumption. *IEEE Internet Computing*. **7** (2) 88-90.

VINOSKI, S. 2004. Is your middleware dead? *IEEE Internet Computing*. **8** (5) 94-96.

VISSER, N. 2003. Identifying the fastest method of online customer related data storage and retrieval between two storage media: Database and Flat Files. *Technologiae*. **1** (2003) 1-9.

WACLENA, K. 1997. *Introduction to the Tcl Programming Language* [Online]. Available from: http://www.lib.uchicago.edu/keith/tcl-course/ [Accessed: 06/02/2004].

WESSON, J & van GREUNEN, D. 2002. Visualisation of Usability Data: Measuring Task Efficiency. In: *Proceedings of the annual research conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT) held in Port Elizabeth on 16-18 September 2002*. Pretoria: UNISA Press: 11-18.

WESTMAN, S. 2002. Building database-backed Web applications: Process and issues. *Information Technology and Libraries*. **21** (2) 63-73.

WHITE, C.J. 2003. *Research: An introduction for educators*. Pretoria: Pierre Van Ryneveld.

WIEDENBECK, S., RAMALINGAM, V., SARASAMMA, S. & CORRITORE, C.L. 1999. A comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting with Computers*. **11** (3) 252-282.

WIGGLESWORTH, J. & LUMBY, P. 2000. *Java programming: advanced topics*. Cambridge: Thomson Learning.

WIRTH, N. 1993. Recollections about the development of Pascal. *ACM SIGPLAN Notices*. **28** (3) 333-342.

YAGER, T. 2001. Web Scripting language: Are you down with PHP?, *InfoWorld*. **23** (50).

YANK, K. 2001. *Which Server-Side Language is Right for You?* [Online]. Available from: http://www.sitepoint.com/print/server-side-language-right [Accessed: 20/10/2004].

YANK, K. 2003. *Review – PHPEd 3.2 (NuSphere)* [Online]. Available from: http://www.sitepoint.com/print/review-phped-3-2-nusphere [Accessed: 20/10/2004].

YE, N. 2001. Robust intrusion tolerance in information systems. *Information Management & Computer Security*. **9** (1) 38-43.

YERKEY, N. 2001. Active Server Pages for dynamic database Web access. *Library High Tech*. **19** (2) 133-142.

ZAK, D. 2001. *An introduction to Programming with C++*. 2nd edition. Boston, Mass.: Thomson Learning.

9 Other bibliographies

CORBETTA, P. 2003. *Social research: theory, methods and techniques*. London: SAGE.

DUBINSKY, Y. & HASSAN, O. 2005. A framework for teaching software development methods. *Computer Science Education*. **15** (4) 275-296.

FRANTZ, R. 1999. Java Servlets: An alternative to CGI scripts, *Byte*, **03605280** (8).

GOMEZ-ALBARRAN, M. 2005. The teaching and Learning of programming: A survey of supporting software tools. *The Computer Journal*. **48** (2) 130-144.

KOLLING, M., QUIG, B., PATTERSON, A. & ROSENBERG, J. 2003. The BlueJ system and its pedagogy. *Journal of Computer Science Education*. **12** (4) 249-268.

MILNE, I. & ROWE, G. 2002. Difficulties in learning and teaching programming: Views of students and tutors. *Education and Information Technologies*. **7** (1) 55-66.

MORRIS, C. 2005. *Technical aspects of creating a thesis/dissertation using Microsoft Word*. Pretoria: TUT Press.

MOUTON, J. 2001. *How to succeed in your Master's & Doctoral Studies: A South African Guide and Resource book*. Pretoria: Van Schaik.

RAGONIS, N. & BEN-ARI, M. 2005. A long term investigation of the comprehension of OOP concepts by novices. *Computer Science Education*. **15** (3) 203-221.

SCACH, S.R. 1996. *Classical and object-oriented Software engineering*, 3rd edition. Chicago: Times Mirror Higher Education Group.

SIEBERHAGEN, A. & BIJL, J. 2004. *Citation and bibliographic reference guide*. Pretoria: TUT Press.

VENTURA, P.R. 2005. Identifying predictors of success for an objects-first CSI. *Computer Science Education*. **15** (3) 223-243.

WEINSCHENK, S., JAMAR, P. & YEO, S.C. 1997. *GUI design essentials*. New York: John Willey & Sons.

Appendix 1: Platforms

ARRIVING AT THE CHOICE OF WEB-BASED PLATFORM TO BE USED IN TESTING THE FRAMEWORK

1.0 Introduction

Before testing the developed framework by evaluating various platforms, it is important to first narrow the choice of platforms to be evaluated. While users can test the framework using the platforms of their choice, we find it necessary to give a brief evolution and description of some platforms that are commonly used for teaching and in the industry. We then attribute reasons why some of the platforms are not used in testing the framework, thereby narrowing down our choice of platforms.

a) Common Gateway Interface (CGI): Yerkey (2001) states that CGI is an early server-side processing method, still widely in use today, in which programs written according to the CGI standard take request from a Web page, retrieve and process data from a database, and send the processed data back to the user's browser. CGI programs may be written in almost any language that can be compiled: Visual Basic, C, C++, Perl, and so on (Yerkey, 2001; Sun Microsystems, 2000; Adida, 1997).

At runtime, a CGI program is launched by the Web server as a separate operating system (OS) shell. The shell includes an OS environment and process to execute the code of the CGI program, which resides within the server's file system. However, each new CGI request launches a new operating system shell on the server (Sun Microsystems, 2000).

As adapted from Sun Educational services (2000), CGI programs have four advantages:

- Programs can be written in a variety of languages, although they are primarily written in Perl.
- A buggy CGI program will not crash the Web server.

- Programs are easy for a Web designer to reference.
- Because CGI programs execute in their own O/S shell, they do not have the concurrency conflicts with other HTTP requests.

Sun Educational services (2000) also indicate that CGI programs have some disadvantages:

- The response time of CGI programs is very high, because CGI programs execute in their own OS shell, causing a heavyweight activity for the OS.
- CGI is not scalable. There is a limit on the number of separate operating system processes a computer can run.
- The languages for CGI are not always secure, or object oriented.
- The CGI script has to generate an HTML response, so the CGI code is mingled with HTML. This is not a good separation of presentation and business logic.

However, Frantz (1999) mentioned that although original CGI programs are started every time you invoke a command, and for this and other mentioned reasons, have fallen out of favor, many servers support something called Fast CGI. This allows the program to continue to execute between commands, and avoid the overhead of starting up for each command.

It should be understood that CGI is an interface, and can not be directly programmed; a script or executable program must be executed to interact with it. Deitel *et al* (2001, p.962) explains that Practical Extraction and Report Language (PERL), whose development was started by Larry Wall in 1987 while working at UNISYS, is commonly used with the Common Gateway Interface (CGI) protocol.

Deitel *et al* (2001, p.962) further indicates that PERL is commonly used due to its power and flexibility. This is further highlighted by (Deitel *et al*, 2001, p.963)'s opinion that because most of the information users send to servers is text, PERL was a logical choice for programming the server side of interactive Web-based applications, due to its simple, yet powerful text processing capabilities. However, since PERL is commonly used with the CGI, it will partake in CGI's limitations mentioned earlier.

b) FastCGI: According to Apte *et al* (2003), FastCGI addresses the process creation problem in CGI by having processes that are persistent. This is made possible because the Web server creates a process for serving dynamic requests either at startup or after the first dynamic request arrives. Further requests are sent using a connection to the fastCGI process, and the FastCGI process sends output back on the same connection to the Web server and ultimately to the client browser.

However, despite the improvement by FastCGI, Lim (2002) indicates that both CGI and FastCGI are now only being mentioned on a note of historical importance in Web development evolution. Moreover, According to Westman (2002), though FastCGI provides flexibility and power, it has several drawbacks, namely: it is expensive in terms development time and requires much greater level of developer expertise; it involves more lines of code (with proportionately greater possibility for bugs/errors) and scales poorly for multiple or large projects in terms of development costs.

1.1 Some other dynamic Web platforms

a) Java Servlets: Sun Microsystems (1997) describes Servlet as a body of Java code that is loaded into and runs inside a Servlet engine, such as a Web server. According to Cooper (2001), Servlets are managed by a Servlet container, which is a component of any Web server capable of running Servlets, and manages Internet connections by a method called "service" which has two parameters - a request object and a response object. These receive and respond to requests from clients respectively. For example, a client may need information from a database; a Servlet can be written that receives the request, gets and processes the data as needed by the client, and then returns it to the client.

A Servlet class is created by extending the *HTTPServlet* class of the Java SDK API. The most important methods of the *HTTPServlet* class are the two service methods doGET and doPOST. These respond to GET and POST method connections respectively, and usually, they are over-ridden with identical code, with each having request and response parameters (Cooper, 2001).

According to Sun Educational services (2000), the basic processing steps for Java Servlets are quite similar to the ones for CGI, but Servlet runs as a thread in the Web container instead of in a separate OS process. The Web container itself is an OS process, but it runs as a service and is available continuously as opposed to a CGI script in which a new OS process (and shell) is created for each request. When the number of requests for a Servlet rises, no additional instances of the Servlet or operating system processes are created. Each request is processed concurrently using one Java thread per request.

As adapted from Sun Educational services (2000), Java Servlet programs have the following advantages:

- Each request is run in a separate thread, so Servlet request processing is significantly faster than traditional CGI processing.
- Servlets are scalable. Many more requests can be executed because the Web container uses a thread rather than an operating system process.
- Servlets are robust and object-oriented. One has all the capabilities of the Java programming language when writing the programs.
- Servlets can only be written in the Java programming language, which makes them easy to write if you know the Java programming language.
- Servlets are platform independent, because they are written in the Java programming language.
- The Web container provides additional services to the Servlets, such as error handling and security.

As adapted from Sun Educational services (2000), Servlets also have the following disadvantages:

- Servlets often contain both *business logic* and *presentation logic.* Presentation logic is anything that controls how the application presents information to the user. Generating the HTML response within the Servlet code is presentation logic. Business logic is anything that manipulates data in order to accomplish something, such as storing data. Mixing both *business logic* and *presentation*

logic means that whenever a Web page changes, the Servlet must be re-written, recompiled, and redeployed.

- Servlets must handle concurrency issues.

The disadvantage of mixing both *business logic* and *presentation logic* led to the development of the Model 2 architecture which involved combining Servlets with the JavaServer Pages technology (Sun Educational services, 2000).

b) Java Server Pages (JSP): According to jGuru (2000), while there are numerous technologies for building Web applications that serve dynamic content, the one that has really caught the attention of the development community is Java Server Pages (JSP). And that, he states, is not without ample reason either; JSP not only enjoys cross-platform and cross-Web-server support, but effectively melds the power of server-side Java technology with the WYSIWYG features of static HTML pages. JSP pages typically comprise of static HTML/XML components, special JSP tags, and optionally, snippets of code written in the Java programming language called "scriptlets" (jGuru, 2000).

It is important to note that the JSP specification is a standard extension defined on top of the Servlet API. Thus, it leverages all of the experience with Servlets (jGuru, 2000). Unlike Servlets, which is a programmatic technology requiring significant developer expertise, JSP appeals to a much wider audience. It can be used by both developers and also by page designers, who can now play a more direct role in the development life cycle (jGuru, 2000).

Another advantage of JSP is the inherent separation of presentation from content facilitated by the technology (Clark, 1999), due to its reliance upon reusable component technologies like the JavaBeans component architecture and Enterprise JavaBeans technology (jGuru, 2000). JSP provide a way to write dynamic Web pages as HTML documents, rather than as output from a Servlet. Essentially, a JSP is an HTML document with embedded Java code (Sun Educational services, 2000).

While JSP mixes code with HTML in the same way as Microsoft's ASP, the advantage of JSP over ASP is that developers get the full power of the Java programming model (Clark, 1999), rather than more lightweight scripting language. In addition, ASP pages must be reinterpreted each time they are run, while JSP-generated Servlets can be compiled once and run multiple times (Karpinski, 1998).

Because JSP pages are translated into Java Servlets (Clark, 1999), JSP technology has all the advantages of Servlets; high performance and scalability because threads are used rather than operating system's shells or processes; platform independent, APIs and the object-oriented language benefits (Sun Educational services, 2000).

It offers an opportunity for creating a clear separation between Model-View-Controller (MVC) aspects of Web-based application, taking advantage of JSPs to separate programming tasks from HTML authoring (Clark, 1999). The Servlets act as a controller, and delegate view functionality to HTML pages and JSPs. For MVC, one would create classes or JavaBeans to provide the business or core application logic (jGuru, 2000; Sun Educational services, 2002). jGuru (2000) indicates that JavaBeans components are nothing but Java objects, which follow a well-defined design/naming pattern: the bean encapsulates its properties by declaring them private and provides public accessor (getter/setter) methods for reading and modifying their values

This combination of using Servlet Controllers and JSP page Views provides the most advantages. The Model 2 architecture makes Web applications fast, powerful, easy to create for an accomplished Java technology developer, cross-platform, scalable, and maintainable by supporting separation of presentation and business logic (Sun Educational services, 2002).

JSP technology has the following disadvantages as adapted from Sun Educational services (2002):

- Often JSP pages contain both *business logic* and *presentation logic.*
- JSP pages must also consider concurrency issues.

- JSP pages are difficult to debug.

Moreover, despite the views that JSP is simpler to write than Servlets, Cox (2001) mentioned that it is better to exploit Java's type-checking ability to validate field parameters and detect invalid links between pages.

c) Microsoft's Active Server Pages (ASP): According to Yerkey (2001), ASP is Microsoft's way of imbedding scripting statements directly into the Web page, as in JSP, rather than by using a separate program as in Servlets, and it differs from CGI in the following ways:

- ASP is proprietary to Microsoft and CGI is not. ASP requires the use of Microsoft server software, and works best in a complete Microsoft environment; CGI can be used on almost all computer platforms, and is widely used on UNIX servers.
- ASP usually uses VBScript whereas CGI programs may be written in any language that can be compiled. Although VBScript may be used in client-side processing, it is always server-side in an ASP context.
- ASP imbeds the script within the Web page, while CGI programs stand alone; separate from the Web page that calls them.
- Unless one is a programmer, it is much easier to use ASP than CGI.
- For most application, CGI is slower than ASP.

With regard to components reuse, Yerkey (2001) quoted Kauffman (1999) as stating that:

> ASP uses ActiveX Data Objects (ADO), which is a Microsoft-developed technology for working with databases on the Internet. They are pre-packaged programming libraries that are built into the Web server and called upon to do some work when it needs to be done. ADO relieves the programmer from writing much code for every little thing that he/she wants to do[13].

MS Access has a built-in utility for exporting an Access table, query, form, or report as an AS page. It is limited in functionality, although it may be a good start. One benefit of ASP is that the full power of a relational database is available to the user (Yerkey, 2001).

[13] As an illustration, not much work could be done if a car manufacturer has to manufacture tyres also. Instead, he/she simply buy tyres and fit them. In the same way, ASP programmers can reach into a set of pre-built tools (ADO) and concentrate on the larger picture of working with data (Yerkey, 2001).

Yerkey (2001) however gave one serious limitation with ASP; being that it works best in a complete Microsoft environment. The server must be running Windows NT server 4.0 or better, Windows 2000, or Windows SQL Server with Internet Information Server installed, or Microsoft's Personal Web Server (PWS), available with Windows 95/98 or NT Workstation. However, in the author's opinion, this limitation could also be viewed as an advantage in that, once you buy the complete Windows package, you can install PWS or IIS at no extra cost.

d) Personal Home Page (PHP): Ross, Darryl, Zymaris & Con (2000) states that although initially developed by Rasmus Lerdorf, PHP is now maintained by a world-wide group of open-source coders. Ross *et al* (2000) also states that PHP is powerful enough to make Perl or C++ coders feel at home, yet it has a syntax that is simple enough for Visual Basic programmers to quickly get up to speed.

The PHP engine can be invoked either as a CGI-callable interpreter from supported UNIX platforms and Windows NT, or as an in-process Apache module, and the PHP scripts, like ASP and JSP, are embedded within HTML code served up by the Web server. The embedded PHP code is executed by the PHP interpreter and any printf()s that it writes are incorporated into the outgoing HTTP stream from Web server to client's browser. Thus, like ASP and JSP, you develop applications that can be viewed by all HTML-capable browsers (Ross *et al*, 2000).

According to Yager (2001), among PHP's strength is its capability for driving just about any database. In addition to MySQL database, PHP directly supports Oracle, Sybase, Informix, SQL Server, and others, as well as databases reachable via ODBC. Also, Yager (2001) states that PHP's easy-to-learn syntax, strong database connectivity, broad platform compatibility, and huge base of contributed extension account for its large and growing base of users.

PHP version 4 came with multithreaded Web server support (which improves performance), buffered output, and an optimized execution engine among many other added features (Yager, 2001). It is free and open-source, so companies can exploit PHP's technical strengths and lower their operating costs at the same time. Whereas, Apache is the Web server of choice for PHP developers, users of other servers needn't switch to use the language (Yager, 2001).

According to Ross *et al* (2000) PHP's syntax can be described as a cross between C, Perl, and shell, and by most accounts, is easier to learn than, say ASP, Python, JSP, or Perl. It also has a wealth of functions – over 700. Like many scripting languages, PHP is typeless, and variable type is decided by the interpreter at run time (Ross *et al*, 2000).

Ross et al, 2000) also indicates that PHP offers object-oriented capabilities. Classes can incorporate variables and functions, have constructors, and can be extended.

PHP's arrays are powerful, and can act as associative arrays (hash tables) and standard indexed arrays. They also hold data of different types and can be nested to arbitrary levels. Thus, one can use them to build complex data structures, and there are numerous functions for manipulating the arrays and stepping through the data (Ross *et al*, 2000).

Furthermore, Yager (2001) states that there are few roles PHP, aided by its many extensions, can't fulfill, though object-oriented programming adherents may point to PHP's limited object support by missing destructors, private members and multiple inheritance, among other things, as limiting PHP's usefulness in complex projects.

However, Yager (2001) states that PHP lacks inherent support for transactions, but it can handle SQL transactions, as long as all the transaction code fits on a single Web page. And although there are XML-RPC and SOAP (Simple Object Access Protocol) extensions, PHP can't transform objects into Web services as effortless as .NET can. Yet, Yager (2001) indicates that as long as architects and coders can work within and around these constraints, PHP deserves consideration as a low-cost, quick-turnaround alternative

to ASP and Java. Finally, though it suffers a few enterprise limitations compared to Java and ASP, but its ease of use, low cost, and rich library of contributed code makes it worth the effort to work around its few shortcomings (Yager, 2001).

e) Python: Python Software Foundation (2004) explains that Python is a product of the Python Software Foundation, which is a non-profit organization. It is an interpreted, interactive, object-oriented programming language like Tcl, Perl or Java. Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic binding. The Python implementation is portable; it runs on many brands of Unix, on Windows, OS/2, Mac, Amiga, and many other platforms (Python Software Foundation, 2004).

According to van Rossum & Drake (2003), Python offers much more error checking than C, and being a very-high-level language, it has high-level data types built in, such as flexible arrays and dictionaries that would cost you days to implement efficiently in C. It is thus applicable to a much larger problem domain than Awk or even Perl, yet many things are at least as easy in Python as in those languages.

It is further stated by van Rossum & Drake (2003) that being an interpreted language, you can save considerable time during program development because no compilation and linking is necessary, giving a fast write/compile/test/re-compile cycle. This also makes it easy to experiment with features of the language, to write throw-away programs, or to test functions during bottom-up program development.

Moreover, in terms program maintenance, van Rossum & Drake (2003) indicate that Python allows writing very compact and readable programs, and that, programs written in Python are typically much shorter than equivalent C or C++ programs, for several reasons:

- High-level data types allow one to express complex operations in a single statement;

- Statement grouping is done by indentation instead of beginning and ending brackets;
- No variable or argument declarations are necessary.

Finally, van Rossum & Drake (2003) states that the language is named after the BBC show "Monty Python's Flying Circus" and has nothing to do with nasty reptiles. According to Python Software Foundation (2003), most of the Python developers seem to be on Unix, though there are a number of Windows and MacOS programmers.

f) Microsoft's ASP.NET: Ervin (2000) states that among the various interactive uses of the Web, database access has, until recently, been a cross-platform, multi-language, multi-interface endeavor not suited to the faint-of-heart. Fortunately, Microsoft's ever-increasing domination of the software industry has led to the consolidation of many tools in one application (Ervin, 2000).

According to MSDN (2002) the Microsoft.NET Framework represents a major change in the way Web applications are built and run. Microsoft ASP.NET is one of the numerous technologies that are part of the Microsoft.NET Framework. The goal of the Microsoft.NET platform is to simplify Web development (MSDN, 2002). Also, according to MSDN (2002) the .NET platform spans clients, servers, and services, and it consists of:

- A programming model that enables developers to build Extensible Markup Language (XML) Web services and applications.
- A set of building block services that are a user-centric set of XML Web services that move control of user data from applications to users. An example is the Microsoft Passport that makes it easier to integrate various applications.
- A set of .NET Enterprise Servers, including Windows 2000, Microsoft SQL Server, and Microsoft BizTalk Server, that integrate, run, operate and manage XML Web services and applications.
- Tools, such as the Visual Studio .NET, which can be used to develop XML Web services and Windows and Web applications for an enriched user experience.

As indicated by MSDN (2002), the unified nature of the .NET framework means that all applications, whether they are Windows applications, Web applications, or XML Web services, are developed by using a common set of tools and code, and are easily integrated with one another. MSDN (2002) further states that the .NET framework also contains:

- The common language runtime, which handles runtime services, including language integration, security and memory management.
- Class libraries, which provide reusable code for most common tasks, including data access, XML Web service development, and Web and Windows Forms.

In summary, what problems does .NET solve? MSDN (2002) indicates that it solves the inability of applications to seamlessly interact with other applications. Developers spend majority of their time rewriting applications to work on each type of platform and client, rather than spending their time designing new applications. The .NET framework solves the problems by providing the runtime, which is language-independent and platform independent, and by making use of the industry-standard XML. Language independence in .NET allows developers to build an application in any .NET-based language (about 27 including Java, c# *etc*.) and know that the Web application will work on any client that supports .NET. Moreover, XML, as an open standard, implies that most modern clients, such as computer operating systems, cellular telephones, personal digital assistants (PDAs), and game consoles, can accept XML data (MSDN, 2002).

g) Tcl: Tcl is a general purpose programming language that was originally intended to be embedded in other applications as a configuration and extension language. One of its most important embeddings, the Tk toolkit, has been successful for the Windows system, and thereby resulted in Tcl and Tk together being most heavily used for building graphical user interfaces, and as a scripting language like Awk or Rexx (Waclena, 1997).

According to Apache (2004), Tcl is being used for the following reasons and features:

- Flexibility: One can program in a style that suits one's needs, either quick and dirty, mixing up HTML and Tcl script to create dynamic Web Web pages fast, or using more organization for larger, complex site with three-tier model.
- Multi-Purpose: It can be used for the Web, but it was also designed as a general-purpose "Tool Command Language" for a variety of other tasks including system administration, rapid GUI creation.
- Wide Industry Use: It has a liberal Open Source license (BSD), which has helped spread it within the industry.
- Fast and Light: Tcl is fast, having undergone major improvements. System resources are also impacted lightly, as Tcl takes up relatively small amounts of disk and memory space.
- Easy: Its simple syntax, and lots of documentation, both in print and on the Web, makes it an easy language to learn, both for the experienced software professional as well as well as for students, Web authors, hobbyists and others.
- Availability of existing code: There are thousands upon thousands of lines of existing code to be used, either as examples to be learned from, to be included directly in your code, or as extensions to perform a range of common tasks.

Wclena (1997) agrees that Tcl is unique in the combination of features it offers. More features given include:

- Embeddability: It's a small language that is designed to be embedded in other applications (such as C, Tk) as a configuration and extension language. This minimizes the number of languages that programmers need to learn in order to configure their applications.
- Extensibility: It is easy to extend the language by the addition of new primitives in C, which has led to many useful extensions such as DBMS access, SNMP etc.
- Platform portability: Tcl is a powerful scripting language that runs under Unix, Linux, VMS, DOS/Windows, OS/2, and MacOS (at least).
- Automatic Memory management: The programmer never needs to allocate memory or specify maximum sizes since all Tcl data structures are dynamically allocated and fully variable in size.

- High-level programming features: It provides all the usual high-level programming features such as variable-length strings, associative arrays, lists, structures or records, ability to redefine procedures at run-time and error handling.

Wclena (1997) further indicated that Tcl is a successful language with the developer community now in the tens of thousands with thousands of Tcl applications in existence or under development. In fact, it mentioned that Tcl is the programming language of the University of Chicago Phoenix Project, and the designer of Tcl/Tk, John Ousterhout, has recently moved to Sun Microsystems from UC Berkeley to head up a group dedicated to Tck/Tk. This, Wclena (1997) states could lead to more interactions of Tck/Tk to other programming languages like Java.

h) Coldfussion: According to Puck (2004), ColdFusion is another popular server-side scripting language, and the features of Macromedia ColdFusion MX include:

- Easy XML integration: Easily integrates XML data using built-in XML document parsing, XSL transformations, and automated serialization of data into XML.
- Advanced Development Capabilities: Create self-documenting, reusable application components callable from CFML pages or as Web services.
- Comprehensive tools support: Rapidly design HTML pages and code server-logic with the tightly integrated visual development tools in Dreamweaver MX.
- High performance architecture: Ensure high performance and reliability by leveraging the full power of the Java technology platform.
- Advanced Server management: Effortlessly and reliably package, deploy, archive, or restore entire applications in a single step.
- Enterprise system integration: Connect to virtually any data source with drivers.
- Flexible application development: Develop and deploy ColdFusion applications on leading J2EE application servers, including Macromedia Jrun, IBM WebSphere Application Server, and Sun ONE Application Server.

However, one major limitation of ColdFusion is that it is only available free on a 30-day trial after which one has to purchase it (Puck, 2004). This may limit the access of students.

Appendix 2: Survey

THE DYNAMIC WEB PLATFORMS SURVEY

1.0 Introduction

Towards the goal of developing a framework for determining suitable platform for teaching Web application development, we have established various criteria. We have also conducted experiment to determine the latency of the Web platforms so as to give weight to platforms with high performance. However, we agree with Dix *et al* (1998, p.431) which states that the best way to find out how a system meets users' requirements is to 'ask the user'. It is therefore important that we receive feedback regarding their impression on the Web application development platforms.

Therefore, in this section, we would present the result of a survey to study or find out from Web application developers / students how they find the use of the Web-based dynamic middleware systems in their daily works. Survey was used as the query technique to obtain information from respondents. The questionnaire is targeted at Web application development students who have been taught using the 4 Web-based platforms under study.

The information requested relates to the features and characteristics of four Web-based middleware platforms namely Java Servlets, JSP, ASP and PHP that are used in the design of applications that runs on the Web. The survey questionnaires contain questions whose answers would yield information like the ease of learning and using the platforms, the percentage of time needed to learn the middleware systems, the percentage frequency of errors, statistics of support services used, suitability for teaching second year students, as well as the overall comparison of the platforms.

1.1 Survey population

The population for the survey is all students and immediate past students of the Tshwane University of Technology, who have been taught using the 4 Web-based platforms under study. In essence, the population amounts to students who specialized or are specializing in Web development applications development stream on the ND: Information Technology degree. There are currently 30 students in the third year and 25 who were the first set graduated in the previous year. This gives a total population of 55 students.

1.2 Survey sampling method

As stated by Corbetta (2003, p.211), sampling is the procedure through which we pick out, from a set of units that make up the object of survey (the population), a limited number of cases (sample) chosen according to criteria that enable the results obtained by studying the sample to be extrapolated to the whole population.

However, as mentioned by Leedy & Omrod (2001, p.102), rather than sample a large number of people with the intent of making generalizations, qualitative research tend to select a few participants who can best shed light on the phenomenon under consideration. The above then suggests that convenient sampling of the population would be more appropriate in this situation.

1.3 Survey data collection

Two final-year IT Diploma students were employed as research assistants to assist the researcher in the collection of the data. A total of 80 questionnaires were sent out. Some were given to the third year Web development students. For the students who recently completed their studies and are busy doing in-service training in the industry, the questionnaires were sent to them to be completed and submitted as part of their in-service training report. A total of 52 completed questionnaires were received thus representing over 90% of the estimated population.

The scalar style of questions was used. This is an adoption of the Likert technique. Corbetta (2003, p.170) stated that (Likert, 1932) proposed the scaling at the beginning of the 1930s. The style expects the user to judge a specific statement on a numeric scale (1 to 5), usually corresponding to a measure of agreement or disagreement with the statements (Corbetta, 2003, p.170), and may be in ascending or descending order of importance. It is perceived better that the most positive response must be mapped with the highest digit on the scale. These values can then be summed to obtain a measure of agreement or suitability for each Web-based dynamic middleware system.

Though, open-ended questions give users an unrestricted or unbounded option of stating issues the way they see it, it was not used in this study because of the problem associated with the corresponding analysis since users' opinions may vary widely.

1.4 Survey hypothesis

The null hypothesis is that there is no significant difference in the respondents' perception of each of the Web based dynamic middleware systems. The alternate hypothesis is that there is significant difference in the respondents' perception of each of the Web based dynamic middleware systems.

1.5 Survey data analysis

According to Corbetta (2003, p.42), data analysis is perhaps the phase in which the difference between the quantitative and qualitative approach is most visible. The difference is further illustrated as follows:

> Quantitative research makes ample use of mathematical and statistical tools, together with a whole array of tables, graphs, statistical tests, etc., as well as the full set of technological equipment (computers, files, data banks, software, etc.). The impact of these weaponry contrasts starkly with the sobriety of a qualitative analysis, in which there is no statistical-mathematical apparatus and the contribution of information technology (if any) is limited to the organization of empirical material. (Corbetta, 2003, p.42-43)

The analysis for the survey data was done using simple statistical parametric analysis, such as frequency distributions and means. The mean values for all characteristics were summed up for each platform to obtain a total score. Pictorial charts were also used.

1.6 Establishing reliability of the survey

According to Coertze and Heath (1997), a reliable scale or instrument e.g. questionnaire is one that is consistent and reliable. It should produce more or less the same accurate results every time it is applied, even when applied by different persons.

As a way to increase the validity of this study, the research assistants were trained before starting to collect data. Also, various sources agree that a way to increase reliability is to be clear and unambiguous, so that different people will agree on the meaning of each item. To achieve this, the questionnaires were piloted on some computer lecturing staff in the Soshanguve campus, to ensure it is clear and unambiguous.

1.7 Establishing validity of the survey

According to Coertze and Heath (1997), validity is concerned with soundness or the effectiveness of the measuring instrument. Validity would raise questions such as:
Does the test measure what it is supposed to measure? How well? How comprehensively? How accurately does it measure it? (Coertze and Heath, 1997).

To increase the validity of this study, the questionnaires were administered to only respondents in the computing and information technology fields. Sampling groups are chosen to include only the practicing programmers in Web applications using Web-based dynamic middleware systems.

The study was concluded within the time frame of 6 months from data collection to report writing, as information obtained after such period may no longer be valid. This is because technologies in the computing world are changing rapidly. White (2003) stated

that reliability can also be seen as an integral part of validity. Therefore ensuring reliability also contributes to validity.

1.8 Scope of the survey

It is very important that sampling groups were chosen to include respondents who have been taught or who have learnt all the four platforms. Otherwise, a respondent may be biased in favor of the particular platform he or she knows.

The study covered a minimum period of six months within which meaningful work can be done while at the same time not too long for the current middleware systems to become obsolete. The investigations were limited to the four dynamic Web platforms evaluated earlier, namely; Java Servlets, JSP, ASP, and PHP.

1.9 Assumptions on the survey

It is assumed that respondents have good working knowledge of programming using middleware platforms. Therefore, they did not just study each platform, but they are equally good in developing applications using each of the middleware platforms.

1.10 Ethical considerations on the survey

The research assistants were trained on data collection and research ethics. They were therefore required to enforce ethical considerations while administering the questionnaires to respondents to collect the data. The respondents were given the choice to remain anonymous or to give their personal information. In such cases where their personal information was given, they were assured that such information would not be revealed without their consent.

1.11 Survey results

Scores were allocated to responses from the respondents, and the average values given by all respondents were summed up for each middleware platform.

Table 1.1. Average "1 to 5" scoring for the platforms based on respondents' perception of the characteristics and attributes desirable.

		Servlet	JSP	ASP	PHP
1	Estimated percentage of task completed per unit time	3.2	4.05	4.05	4.9
2	Estimated percentage of commands used out of the total available commands while still able to complete a working application?:	4.15	4.05	4.85	4.8
3	Estimated percentage number of good features available	4.2	5.0	4.3	4.75
4	Estimated percentage of times that you express satisfaction	3.95	4.95	4.8	4.9
5	Estimated percentage of times that the online help/ documentation proved very useful:	3.25	3.95	3.75	3.95
6	Estimated percentage of times that you easily recover from a runtime error situation?	4.1	4.95	4.95	4.65
7	Estimated percentage of times that errors are clearly explained and error-exit provided	3.2	2.95	3.8	3.95
8	What is the estimated percentage of the middleware system that you can comprehend and start using effectively within a week?	3.25	3.95	4.95	4.8
9	Estimated percentage of commands/actions in the system is similar to those of other software common to your operating system?	3.7	3.95	4.8	3.85
10	Estimated percentage of commands /actions in the system mirrors the actions and so can be remembered easily without memorizing	3.9	3.95	3.85	4.1
11	Estimated percentage of commands/actions in the system that are high level commands or functions?	3.1	3.95	4.75	4.95
12	Suitability for teaching 2nd year Web development students	1.1	2.05	3.9	4.2
13	The ease of use the each middleware platform	3.2	4.05	3.8	4.8
14	Suitability for complex jobs in the Web development industry	4.85	4.0	3.2	3.2
15	Overall comparison of the platforms	3.3	4.0	3.1	4.75
	TOTAL	52.4	59.8	62.8	66.5

On a scale of 1 to 5, where 5="Strongly agree" and 1= "Strongly disagree", table 1.1 above gives the average scoring for the middleware platforms based on the respondents' perception of the characteristics and attributes desirable for the dynamic Web platforms.

The total score on "1 to 5" scale for the platforms is depicted in the figure 1.1 below. From figure 1.1 above, it is clear that PHP is considered most suitable by the respondents. This is followed by ASP. JSP is ranked 3rd while Servlet is ranked lowest.

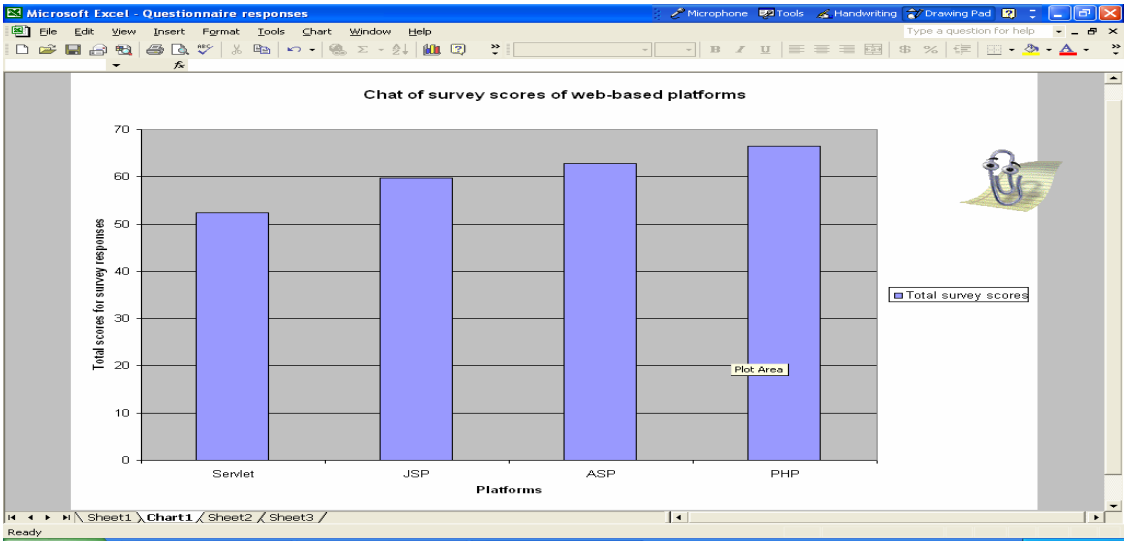

Figure 1.1. Total of "1 to 5" scores.

PHP's highest score could be explained as due to the fact that it is shipped along with MySQL which makes it easier for programmers to implement their Web-based database application systems. ASP's 2nd rank could be due to the similarity of the Microsoft environment. JSP's 3rd rank could be due to the fact that it draws some strength from the Java programming language while adopting the ASP's scripting approach that makes it usable by both programmers and Web designers. Servlet's lowest rank could be due to its low-level syntax, low readability of code, and the problems identifying errors and in debugging the code.

1.12 Conclusion

As indicated earlier, this study involves using different approaches in order to arrive at our conclusions. The survey agrees to a great extent with the results of the testing the framework in this study. This serves to affirm the results of the use and implementation of our developed framework to determine the most suitable platform for teaching Web application development in tertiary institutions in South Africa.