UNIVERSITI PUTRA MALAYSIA

# EVALUATING LOCAL FIRST PAGE UNIVERSITY SITE USING 'U' TOOL

ZALILAH BT. ABD AZIZ

FSKTM 2003 11

# EVALUATING LOCAL FIRST PAGE UNIVERSITY SITE USING 'U' TOOL

By

ZALILAH BT. ABD AZIZ
(GS 08950)

Thesis Submitted to the School of Graduate Studies, Universiti Putra, Malaysia in Partial Fulfillment of the Requirements for the Degree of Master of Science

November 2003

UPM

## ABSTRACT

**EVALUATING LOCAL FIRST PAGE UNIVERSITY SITE USING 'U' TOOL**

**By**

**ZALILAH BT ABD AZIZ (GS 08950)**

Making a web site usable is necessary for a web-page to be successful and for users to be satisfied. Therefore, developing sites that are responsive to user needs is critical for all site designers and managers. With the increasing growth of e-learning and emphasis on higher education in our country, a good and usable university web sites are required for this demand.

This project will focus at evaluating the usability characteristics of every first page of our local universities. The concentration of this project is to identify the proposed metrics and develop a prototype considering only the automated usability characteristics based on the ISO 9126 standard. Therefore, a prototype of 'U' tool is developed to serve this purpose

## ABSTRAK

**MENILAI MUKA SURAT PERTAMA LAMAN WEB UNIVERSITI TEMPATAN DENGAN MENGGUNAKAN 'U' *TOOL***

**Oleh**

**ZALILAH BT ABD AZIZ (GS 08950)**

Membangunkan suatu laman web yang dilawati dan digunakan adalah penting bagi membuat sesuatu laman berjaya dan pengguna berpuas hati. Maka, pembangunan laman yang responsif kepada kehendak pengguna adalah kritikal kepada semua pembangun laman web dan pengurus. Dengan peningkatan perkembangan e-pembelajaran dan penekanan terhadap pendidikan tinggi di negara kita, laman web universiti yang baik dan dapat digunakan adalah diperlukan bagi memenuhi permintaan ini.

Penumpuan projek ini adalah untuk menilai ciri-ciri penggunaan bagi muka surat pertama setiap laman web universiti awam yang terdapat di Malaysia. Ianya akan mengenal pasti metrik yang dicadangkan dan membangunkan prototaip yang akan hanya mengambilkira ciri-ciri penggunaan yang boleh

diautomasi berdasarkan piawai ISO 9126. Suatu prototaip 'U' *tool* akan dibangunkan bagi menilai ciri-ciri ini.

## ACKNOWLEDGEMENTS

In the name of Allah, the most merciful and most compassionate. Praise to Allah S.W.T. for granted me strength, courage, patience and inspirations in completing this thesis.

My deepest appreciation and gratitude to my Supervisor, Puan Azrina Kamarudin for her guidance, motivation and support that lead the way in so many aspect of this project. Beside that, I would also like to thank all of my lecturers on delivering their knowledge during my studies in UPM. For the financial support, I am grateful to Universiti Teknologi Mara (UiTM) for the scholarship and study leave.

Special appreciation to my parents, husband and my five children for their patience, loves, prayers, understanding and moral support for making the best of my situation. Sincere thanks to friends and colleagues for sharing problems and experiences throughout the years.

Thank You

Zalilah Abd. Aziz
November 2003

## TABLE OF CONTENT

## LIST OF TABLES

## LIST OF FIGURES

# Chapter 1

## INTRODUCTION

### 1.1 Introduction

Increasing emphasis is being placed on good University Web site design, especially on the importance of making the site useable. This becomes even more significant with the growing acceptance of e-Learning. Web-based Information systems (WIS) are growing at a rapid pace, both in terms of the increasing acceptability of web sites, and in terms of the complexity of such products. However, a much defined product process model that leverage the effective development, and the evaluation process model that promote the Web-site quality assessment and improvement are not accompanied by that sites growth [1].

Therefore, a systematic and disciplined utilization of engineering methods, models and techniques for the understanding, assessment and improvement of this kind of software should considered a mandatory requirement. One of the primary goal for University Web-site quantitative evaluation is to understand the extend which a given collection of quality characteristics fulfills a selected set of needs regarding a specific user view. Another aim for University Web sites is to evaluate the level of accomplishment of required

characteristics such as usability, functionality, reliability, performance and accessibility. Developing sites that are responsive to user needs is critical for all site designers and managers (Price 1997). For web-page owners to be successful and for users to be satisfied, web-sites need to consider usability and other design criteria (Nielsen 2000, Pearrow 2000, Shneiderman 1998).

The software measurement can be carried out on different perspective of the software development process such as process measurement, products measurement and resource measurement. Concentration of this project is to focus on the external product attributes, which, in turn, are measured in relation to how the product interacts with other entities in the environment.

There are many ways to formally evaluate the usability of a Web site. Heuristics (design principles) can be used by experts to judge usability (Nielsen, 1993). Benchmarking can be used to compare one web-site with another, or against a set of standards (Misic and Johnson, 1999). A web-site can be evaluated against a checklist of usability items (Keevil, 1998). Prototyping can be used to quickly and cheaply develop a mock site that can be shown to users before the real site is launched. Users can also participate in focus groups to provide feedback on the usability of a site, or can provide data through controlled laboratory sessions (Kantner and Rosenbaum, 1997). Often, usability data is collected through metrics such as the amount of time

that it takes to accomplish a task, or the number of errors that a user makes while searching for information.

## 1.2 Problem Statement

Evaluation methods and techniques can be categorized in qualitative and quantitative. Even though software assessment has more than three decades as a discipline, the systematic and quantitative quality evaluation of Hypermedia applications and in particular the evaluation of web-sites is rather a recent and frequently neglected issue[2]

Measurement of web sites is essential to determine the level of quality at any particular stage in the life cycle, to motivate improvements and to determine whether these have been successfully achieved.

Increasing emphasis is being placed on good web-site design, especially on the importance of making a site useable. Organizations want to ensure that users not only come to their web-site, but also complete their intended tasks in a minimum amount of time. Sometimes users need to find pieces of information. If the site is not easy to use, users may become frustrated and leave before their objectives are accomplished. Web-sites developers also

want users to stay at their Web sites as long as possible, and ultimately come back to them again.

## 1.3 Objective

The main objective of this project are as follows:

- To assist local web sites developers to come out with quality web sites.
- Propose a suitable metric for measuring usability characteristics for local universities web-sites.
- Develop a prototype of automated tools to measure the usability characteristics for local universities web-sites.

## 1.4 Scope

The scope of this project is to focus on measuring the usability characteristics of 10 local universities in Malaysia. The universities selected are as follows:

i. Universiti Utara Malaysia

ii. Universiti Kebangsaan Malaysia

iii. Universiti Sains Malaysia

iv. Universiti Teknologi Malaysia

v. Universiti Islam Antarabangsa

vi. Universiti Teknologi Mara

vii. Universiti Malaysia Sabah

viii. Universiti Malaysia Serawak

ix. Universiti Perguruan Sultan Idris.

x. Universiti Putra Malaysia

The reason for selecting the local universities is we wish to help the local universities web-sites developers to produce quality web-sites in order to fulfill our Government's future aim in promoting Malaysia as an alternative destination for those who wish to get a higher education [Business Trends, March 2003].

All of these web-sites are developed by using the Hypertext Markup language (HTML). The prototype that will be developed using the Active Server Page Language (ASP). This choice was made because the same ASP document may be viewed by many different "browsers", of very different abilities.

This project will only discuss the usability factor of university web sites. Usability is defined as a set of attributes that bear on the effort needed for the use and on the individual assessment of such use by a stated or implied set of users. The methodology chosen will only measure the usability factor that can be automated and was chosen according to literature review done[ Olsina, Godoy, Lafuante and Rossi, 1999].

Among the attributes that will be measured are:

**1. Usability**
1.1 Global Site Understandability
1.1.1 Global Organization Scheme
1.1.1.1 *Site Map*
1.1.1.2 *Table of Content*
1.2 On-line Feedback and Help Features
1.2.1 Quality of Help Features
1.2.1.2 *Search Help*
1.2.2 Web-site Last Update Indicator
1.2.3 Addresses Directory
1.2.3.1 *E-mail Directory*
1.2.3.2 *Phone-Fax Directory*
1.2.3.3 *Post mail Directory*
1.2.4 *FAQ Feature*
1.2.5 On-line Feedback
1.3 Miscellaneous Features
1.3.1 *Foreign Language Support*
1.3.2 *What's New Feature*
1.3.3 *Screen Resolution Indicator*

Only first page of every web-sites is evaluated to fulfill the usability characteristics that is the learnability characteristics. The metric will be calculated and compared among the web-sites. A rating level will be given at the end of the measurement.

# Chapter 2

## LITERATURE REVIEW

### 2.1 Software Measurement

Measurement permeates everyday life and is an essential part in every scientific and engineering discipline. Measurement allows the acquisition of information that can be used for developing theories and models, and devising, assessing, and using methods and techniques. Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules. The rules help us to be consistent in our measurement, as well as providing a basis for interpreting data [FEN96]. Software metric helps managers gather data to better manage their software projects.

Even when a project is not in trouble, measurement is not only useful but necessary. Measurement is needed at least for assessing the status of projects, products, processes and resources. Because we do not always know what details a project, it is essential that we measure and record characteristics of good projects as well as bad. We need to document trends, the magnitude of corrective action, and the resulting changes. In other words, we must control our projects, not just run them. Tom DeMarco, a strong supporter of the need

for measurement in software development asserts that:

"You cannot control what you cannot measure." (DeMarco, 1982)

Every measurement action must be motivated by a particular goal or need that is clearly defined and easily understandable. This situation has prompted Tom Gilb to state the Gilb's principle of Fuzzy Targets that:

"Projects without clear goals will not achieve their goals clearly." (Gilb, 1988)

It is difficult to imagine electrical, mechanical and civil engineering without a central role for measurement. Indeed, science and engineering can be neither effective nor practical without measurement. But, measurement has been considered a luxury in software engineering. For most developers projects:

- We fail to set measurement targets for our software products. For example, we promise that the product will be user-friendly, reliable and maintainable without specifying clearly and objectively what these terms mean. As a result, when the project is complete, we cannot tell if we have met our goals.
- We fail to understand and quantify the component costs of software projects. For example, most projects cannot differentiate the cost of design from the cost of coding or testing. Since excessive cost is a frequent complaint from many of our customers, we cannot hope to control costs if we are not measuring the relative components of cost.

- We do not quantify or predict the quality of the products we produce. Thus, we cannot tell a potential user how reliable a product will be in terms of likelihood of failure in a given period of use, or how much work will be needed to port the product to a different machine environment.
- We allow anecdotal technology, without doing a carefully controlled study to determine if the technology is efficient and effective. Most of the time, these materials are not accompanied by reports of the scientific basis for the claims. [FEN96].

## 2.2 Software Metrics

Measurement on a software development process and its product is performed by applying particular 'metrics'. Measuring will normally comprise several metrics, again resulting in several measurements per metric.

A lot of categorizations and examples of metrics can be found in literature, some example are (Pfleeger, 1991; Fenton and Pfleeger, 1996; Grady, 1992):

- Product and process metric
- Objective and subjective metrics
- Direct and indirect metrics
- Explicit and derived metrics
- Absolute and relative metrics

- Dynamic and static metrics
- Predictive and explanatory metrics.

The most common types of metrics are described below:

- **Product and process metric**

A product metric is a measurement of an intermediate or final product of software development, and therefore addresses the output of a software development activity. Examples of such metric are a size metric for the number of requirements, a complexity metric for the software code, etc. Process metrics measure the characteristics of the overall development process, such as number of defects found throughout the process during different kinds of reviews, etc.[Solingen,Berghout,1999].

- **Objective and Subjective Metric**

Objective metrcis are absolute measures taken of the process or product, and count attributes or characteristics in an objective way (Humphrey, 1989), such as number of lines of code, number of faults discovered. These metrics have a fundamental starting point, a natural zero. Subjective metrics are measurements of a process or product that involve human, subjective judgement. Examples of subjective metrics are expected complexity and degree of conformance to coding standards. These measurements are classifications of observations.

- **Direct and Indirect Metrics**

A direct metric is a measure of a process or product characteristics that does not depend on the measurement of any other characteristics. Examples are the number of faults in a product, number of hours spent during certain process. An indirect metric, on the other hand, is a measurement of a process or product characteristics that involves the measurement of one or more characteristics, such as productivity, fault density, etc. An indirect metric always contains a calculation of at least two other metrics.

### 2.2.1 History of Software Metric

To assess the current status of software metrics, and its successes and failures, we need to consider first its history. Although the first dedicated book on software metrics was not published until 1976 [Gilb 1976], the history of active software metrics dates back to the late-1960's. Then the Lines of Code measure (LOC or KLOC for thousands of lines of code) was used routinely as the basis for measuring both programmer productivity (LOC per programmer month) and program quality (defects per KLOC). In other words LOC was being used as a surrogate measure of different notions of program *size*. The early resource prediction models (such as those of [Putnam 1978] and [Boehm 1981]) also used LOC or related metrics like *delivered source instructions* as the key size variable. In 1971 Akiyama [Akiyama 1971] published what we

believe was the first attempt to use metrics for software quality prediction when he proposed a crude regression-based model for module defect density (number of defects per KLOC) in terms of the module size measured in KLOC. In other words he was using KLOC as a surrogate measure for program *complexity*.

The obvious drawbacks of using such a crude measure as LOC as a surrogate measure for such different notions of program size such as effort, functionality, and complexity, were recognised in the mid-1970's. The need for more discriminating measures became especially urgent with the increasing diversity of programming languages. After all, a LOC in an assembly language is not comparable in effort, functionality, or complexity to a LOC in a high-level language. Thus, the decade starting from the mid-1970's saw an explosion of interest in measures of software complexity, pioneered by the likes of [Halstead 1977] and [McCabe 1976]) and measures of size such as function points pioneered by [Albrecht 1979] and later by [Symons 1991] which were intended to be independent of programming language.

Work on extending, validating and refining complexity metrics including applying them to new paradigms such as object oriented languages [Chidamber and Kemerer 1994] has been a dominant feature of academic metrics research up to the present day [Fenton 1991, Zuse 1991].

In addition to work on specific metrics and models, much recent work has focused on meta-level activities, the most notable of which are:

- Work on the mechanics of implementing metrics programs. Two pieces of work stand out in this respect:
    1. The work of [Grady and Caswell 1987] later extended in [Grady 1992] which was the first and most extensive experience report of a company-wide software metrics program. This work contains key guidelines and lessons learned which influenced and inspired many subsequent metrics programs.
    2. The work of Basili, Rombach and colleagues on GQM(Goal-Question Metric) [Basili and Rombach 1988]. By borrowing some simple ideas from the Total Quality Management field, Basili and his colleagues proposed a simple scheme for ensuring that metrics activities were always goal-driven. A metrics program established *without* clear and specific goals and objectives is almost certainly doomed to fail [Hall and Fenton 1997]). Basili's high profile in the community and outstanding communications and technology transfer skills ensured that this important message was subsequently widely accepted and applied. That does not mean it is without its criticisms ([Bache and Neil 1995] and [Hetzel 1993] argue that the inherent top-down approach ignores what is feasible to measure at the

bottom). However, most metrics programs at least pay lip service to GQM with the result that such programs should in principle be collecting only those metrics which are relevant to the specific goals.

- The use of metrics in empirical software engineering: specifically we refer to empirical work concerned with evaluating the effectiveness of specific software engineering methods, tools and technologies. This is a great challenge for the academic/research software metrics community. There is now widespread awareness that we can no longer rely purely on the anecdotal claims of self-appointed experts about which new methods really work and how. Increasingly we are seeing measurement studies that quantify the effectiveness of methods and tools. Basili and his colleagues have again pioneered this work (see, for example [Basili and Reiter 1981, Basili *et al* 1986]. Success here is judged by acceptance of empirical results, and the ability to repeat experiments to independently validate results.
- work on theoretical underpinnings of software metrics. This work (exemplified by [Briand et al 1996, Fenton 1991, Zuse 1991] is concerned with improving the level of rigour in the discipline as a whole. For example, there has been considerable work in establishing a measurement theory basis for software metrics activities. The most important success from this work has been the acceptance of the need