IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4921

Measuring Quality Metrics for Web Applications

Ben Lilburne
School of Computing and Information Technology, University of Western Sydney, Locked Bag 1797 Penrith DC, NSW 1797, Australia

Prajwol Devkota
School of Computing and Information Technology, University of Western Sydney, Locked Bag 1797 Penrith DC, NSW 1797, Australia

Khaled Md. Khan
School of Computing and Information Technology, University of Western Sydney, Locked Bag 1797 Penrith DC, NSW 1797 Australia, k.khan@uws.edu.au

ABSTRACT

This paper makes an attempt to propose a framework for measuring quality attributes of web-based application systems. This paper is particularly interested in three major quality attributes: usability and reliability from users point of view; and maintainability for the developer's perspective. The diverse nature of web applications makes it difficult to measure these using existing quality measurement models. Web applications often use large numbers of reusable components which make traditional measurement models less relevant.

INTRODUCTION

The quality of a web application could be measured from two perspectives: quality perceived by the programmers, and quality experienced by the end-users. A website with thousands of manually maintained static pages, each providing similar content, will not have the same degree of maintainability as a site providing the same content through a database and generated pages. In a traditional non web-based system, if the usability and reliability are poor, the user will still have to use it hoping that the next release of the software would fix the problem. In a web-based system, users will simply leave the site and go elsewhere once they find it unacceptable.

Web-based systems have to be maintained constantly to keep their functionality and contents up to date. Web applications have several features that make traditional software quality metrics less effective in producing realistic quality measurements. Web applications tend to be pieced together using a number of components, scripts, static mark-up pages, templates and media files.

Broadly, we can form several categories for contents on the web such as static, semi static, and dynamic in terms of web pages and media. Static pages are simple mark-up, with no dynamic content on them. Media images such as audio, video, special effects are considered to be static media. Semi static pages or media can be generated from templates, but without using any significant programming. This includes template languages such as XSLT, XML [2] and similar others. This also includes some client side scripting, such as JavaScript for rollover effects. Dynamic contents are generated as requested using server side programs, or non-trivial client side scripting.

The rates of change in the web format, computational logic and contents may vary considerably, and these are heavily dependent on the domain of the web application and business logic. These changes have considerable effect on the usability, reliability and maintainability of web applications. Our approach is based on the classical factor-criteria-metrics model [7] proposed for non web-based software systems.

The paper proceeds as follows. In section 2, we cite a brief discussion on related research work in this area. Maintainability, reliability and usability are further spelled out in terms of their assessment criteria and sub factors in section 3. Section 4 presents an assessment model for the web application. The paper finally concludes in section 5.

RELATED WORK

Although research on measuring usability, reliability, and maintainability of web-based systems is seriously underrepresented in current software engineering literature, there are some research efforts reported recently. Olsina et al. [3] describes a quality evaluation method for academic websites, which involved a case study on several university websites The research effort only addresses user's perspective to web-based systems. The measurements are based on the documents provided over the web to the browser, with no mention of how these documents may be delivered.

Current metrics for websites seem to be concentrated on counts of pages, links and media files. The work reported in [4] uses reused media counts to measure the reusability of a site and measures based on internal links to measure complexity. The approach in [5] computes size and effort metrics based on the number of components, with various categories of components. The intrinsic quality of the atomic components has not been taken under consideration in the metrics calculation.

Many software quality factors have already been defined and some authors have defined quality factor for web. Boehm's software quality model [1], McCall's [7] software quality models are some early and widely used models. Some web quality factors have already been proposed in [3] and [8].

WEB QUALITY FACTORS

Today web-based software is diverse. Different consumers can have different definitions of quality for the same web-based product. A fixed model is not suitable to address all of the quality requirements of web applications, since the requirements vary for different kinds of sites. Our aim is to propose a simple web quality model that is most common to all web-based software. Then let the software professionals and users tune and redefine the model according to their needs. To specify a quality goal, we use Factor-Criteria-Metrics structure [7] as shown:

```
<a_Quality_Goal>
    <a_set_of_Criteria_for_the_Goal>
        <a_set_of_Metrics>
```

A *factor*, typically a high level quality goal, is measured in terms of a set of *criteria*. Each criterion can then be quantitatively measured

against a set of *metrics*. We propose new quality criteria for maintainability, reliability and usability that we believe are important for web-based software.

Maintainability

It is already well established that a website should be treated as a set of components. Our interest is to consider the nature of these components, and how they affect the web site's quality. Web sites differ from most software systems in a number of ways. They are changed and updated constantly after they are first developed. As a result of this, almost all of the effort involved in running a web site is maintenance. We will use the following criteria for estimating maintainability (from ISO 9126):

- Analysability
- Changeability
- Scalability
- Stability
- Testability

We now look at each of these criteria and their metrics in the following subsections.

Analysability

Weinberg [6] identifies one of the primary problems when analysing a computer program is locality. Locality is defined as how the related components close together in the structure of a program, and this has a lot of influence on how easily a web site is analysed. Because a web site has a number of interrelated components, some of which perform one task as a unit (an HTML form and a CGI to process it, then another static HTML page that has a 'thank you' note for example). We will use the number of components used to perform a task as a measure of inter-component locality in a web application. The analysability of web application can be measured by using the following methods:

- *Locality (L):* Average number of components per task. 1/1 (100%) is the best ratio here. As more components are used to perform a single task, the locality score decreases.
- *Error reporting (ER):* The error reporting score is calculated using Table 1. Starting with 0 and adding the given amount for each point we have.
- *Style consistency (SC):* Components with consistent style to total components, best ratio is 1/1 (100%). Any components that are inconsistent in style will reduce this score.

The total score for analysability is calculated as

$$\frac{L + ER + SC}{3}$$

Changeability

For *changeability* we are interested in how easily the data, formatting and program logic in the website can be changed. This can be measured with the following metrics:

- *Dynamic data ratio (DDR):* The proportion of the data that is generated by programs on the server side, including data that is extracted from a database. It is calculated as

$$\frac{Dynamic\ pages}{total\ pages}$$

- *Dynamic format ratio (DFR):* The proportion of the format that is generated from specific formatting modules, such as templates or a library of format function. The calculation is simple:

$$\frac{Number\ of\ templates}{Total\ number\ of\ formats}$$

- *Pages/format ratio (PFR):* The number of pages presented on the client side that follow a specific defined format. A percentage score for this can be calculated as:

$$\frac{number\ of\ format}{number\ of\ pages} \times 100$$

- *Pages/data ratio (PDR):* The proportion of pages presented on the client side that use the same data or present similar forms of data. This can be computed as:

$$\frac{number\ of\ pages\ with\ similar\ data}{number\ of\ pages} \times 100$$

- *Program/documentation score (PS):* A percentage score aver aged over all of the programs in the web site, where each program is reviewed using either peer review or some other program quality assessment methods. The review should of course be with the perspective of how easy the program logic is to change.
- *Data change rate:* This is a percentage score indicating how important changes in the data will be.
- *Format change rate:* This is a percentage score indicating how important formatting changes will be.

The dynamic data and format ratios are used to determine the current amount of data provided by the site that is dynamically generated. This ratio is then adjusted by the change rates and page ratios as:

$$1 - \frac{((DDR \times PDR) + (DFR \times PFR) + PS)}{2} \times 100$$

which is the average of the program score and the changeability of data and formats. The overall significance of this changeability number should be weighted with the averages of the expected change rates, so a "brochure" style site will not consider changeability as an important criterion.

Problems can occur here when a script can generate an unlimited number of pages. In such a case, an approximate count for the number of pages someone might like to see can be used, or if the output is trivial, such as generated totals on an order form, a count of 1 should be used.

Scalability

Scalability of a web-based software can be defined as the ability to adjust configuration size to fit new conditions and ability to change scaling of an application. Unlike traditional software, web-based system has tendency to grow exponentially. This can get to an unexpected level.

When the number of users increases it is important that the web-based software is able to handle the load or easy to scaled up to handle the load. The hardware platform should also be easy to scale up to cope with the rising load without effecting the operation of the site.

Scalability is an important quality factor for web-based software as it has very high potential to grow unexpectedly. The criteria could be how the system handles an increase number of users.

Table 1: Error Reporting Score

Errors on web pages, no location	0
Errors have location	+25
Errors are kept in a log	+25
Critical errors are e-mailed	+25
Verbose error reports	+25

Stability

Stability is defined as the attributes of a software product that have an influence on the risk of unintended consequences as a result of modifications. There are some specific cases of this sort of consequence in web applications, such as broken links as a result of changing the name of a page. For example, in a highly coupled web system where many links pointing to one page, and if that page is moved or removed, all of the links referring to that page must be changed accordingly. The metrics could be:

Page-page coupling (PPC): This is 1/the number of direct links *to* a page, averaged over all of the pages on the site. Links that are used in multiple pages, but are extracted from the same source (such as a database) should only be counted once.

Testability

Each update to a site made during maintenance should be testable, preferably before the changes go live (online). Since the site can be tested through a web browser exactly like black box testing, it sounds simple. However, testing individual components could be problematic. Most CGI scripts can have values entered manually at a terminal, but the output is made for a browser, not a human.

The varied nature of methods used to test web-based software makes these criteria difficult to measure generally. Rather than using a general rule, we will provide a number of questions that contribute to how testable a web system is. Start at 0 (no testing), and add values according to Table 2.

We have given each of these questions equal weight, however in reality some of these factors are more important than others. In some cases, analysing server logs is the most important part of testing a running system [3]. This scoring could be adjusted according to the importance of the questions in a particular context.

Reliability

Reliability is a very important factor for Web-based systems. We define reliability into two as information reliability and operational reliability. Operational reliability is an important criterion for any e-commerce based web application. However, operational reliability does not apply to information serving web-based systems. Operation and information reliability is a must for all types of web-based systems. There should be no web-based software that allows for intentional operation failure, wrong information and transactional errors.

The major criteria for reliability include consistency, accuracy, completeness, availability and fault tolerance. Consistency, accuracy and completeness address the quality factor information reliability. Whereas, fault tolerance concerns the factor operational reliability. Information reliability is a quality factor which has three criteria: completeness, accuracy, and consistency. Each of these has directly measurable metrics. We propose the following factors, criteria and their metrics for reliability in a hierarchical structure:

```
<Information Reliability> (Factor)
  <Completeness> (Criteria)
   <number of broken links?> (metrics)
  <Accuracy> (Criteria)
    <degree of info. accuracy>(metrics)
  <Consistency> )(Criteria)
    <degree of info. consistency>
<Operational Reliability> (Factor)
  <Availability> (Criteria)
     <site uptime> (metrics)
  <Error tolerance> (Criteria)
     <number of un-trapped errors>
     <number of fault recovered>
```

Table 2: Testability Score

Offline testing available?	20
Previous test results available?	20
Specific testing modes for scripts?	20
Components can be tested individually?	20
Server Log analysis tools available?	20

Usability

Usability of a web-based system is very important for a simple reason - competitiveness. Unlike traditional software, users are not confined to use web sites that they do not feel comfortable with. Users can easily switch to another website that is more easy to use. Usability is therefore considered most important factor of web-based software and application.

Defining a common set of usability criteria is not a good idea as usability requirements for different web sites can be different. Usability requirements of a web site may change from time to time based on the available technology and business competitiveness. For these reasons usability criteria for a web site is best derived from functional specification.

MaCall's software quality model [7] lists operability, training, communicativeness, I/O volume and I/O rate as the usability criteria. Training is not applicable to web sites; it is only applicable to intranet with closed user group. We think availability of 'help' option is more appropriate to all types of web-based software. I/O volume and I/O rate better defined as speed. With some refinement to MaCall's criteria we propose the following criteria for usability:

```
<Usability> (Factor)
  <Operability> (Criteria)
     <easy to use a functionality>
          (metrics)
     <degree of navigability>
          (metrics)
  <Accessibility to help> (Criteria)
     <on-line help/support> (metrics)
  <Communicativeness> (Criteria)
     <easy to comprehend the content>
          (metrics)
  <Speed and space> (criteria)
     <response time> (metrics)
     <memory space required> (metrics)
```

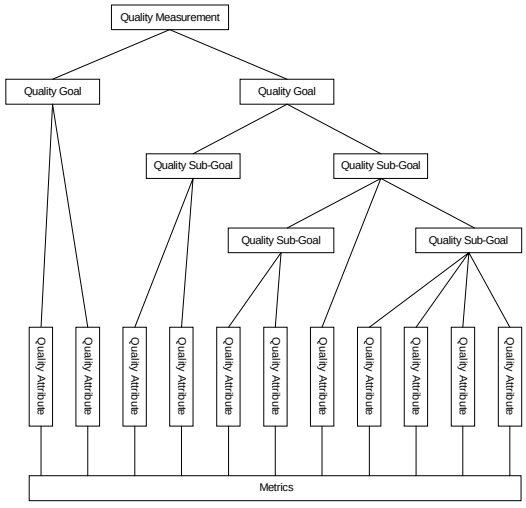
THE FRAMEWORK

In devising our framework we have followed Ejiogu's [11] attributes, and in addition we have included some of our own criteria. As shown in Figure 1, Quality Compliance Framework (QCF) consists of components such as *quality measurement, quality goal (factor), quality sub-goal (criteria),* and *quality attributes (metrics).*

Quality measurement is the quality achievement in terms of a percentage value that indicates the degree of an overall quality compliance of the system. For example, a score of 95% for a system means the system has 95% of quality compliance with the defined quality properties for that system.

Quality goal is a high level quality factor of a system. A quality goal may have many levels of quality sub-goals or quality attributes. Usability is a common example of a quality goal. A system's usability

Figure 1: Quality Compliance Framework(QCF)



of 90% means that the system has 90% compliance with usability goal criteria defined within the quality model of that system.

Quality sub-goal (criteria) is a lower level quality goal that breaks down its parent goal to more measurable goals. A quality sub-goal can have more quality sub-goals or quality attributes. For example, usability may have sub-goal such as operability, learnability and so on.

Quality attribute(metrics) is a measurable unit of quality in QCF. A quality attribute may belong to one or many quality goals or quality sub-goals.

QCF provides the quality measurement in a simple quality compliance scale. The scale starts form 0% and ends at 100%, where 0% indicates no quality compliance and 100% indicates full quality compliance. This is the QCF score of the web system.

QCF works using bottom up approach. The metric for an attribute is converted to a 0% to 100% scale. Then the higher-level QCF score is calculated based on the QCF scores earned by the lower level children attributes, sub-goals, or goals.

Final score is the Quality measurement QCP. Following formulas show how the QCP is calculated for different components of QCF:

- Quality Measurement

$$QualityMeasurement = \frac{\sum childrens'QCF}{No.\_of\_Children}$$

- Goal and sub-goal QCF score

$$GoalQCF\_score = \frac{\sum Childern'sQCF}{No.\_of\_Children}$$

- Attribute QCF score

$$AttributeQCF = \frac{EarnedScore}{PossibleScore} \times 100\%$$

Here, "Children" refers to the Quality goals, Quality sub-goals, or Quality attributes in the hierarchy.

Different phases of software development life cycle will have different set of quality goals, attributes and metrics. QCF allows the separation of these phases but gives one quality measurement of the software at any stage. An important question may be raised when defining quality goals and attributes. How do we stress on important quality goals? It depends on the use context and the type of the web system. To increase the importance of a quality goal, weighting is to be adjusted to the goals in the form of a multiplier to that goal and to decrease the significance of other goals. This means that the simple formulas to calculate average values discussed above will need to be replaced with weighted averages.

QCF can give an early quality feedback throughout the web development process. For example, if there is a drop in overall quality score, anyone can drill down to lower level and find the area where the quality is suffering. Project managers and system owners can use QCF to ensure every phase of software development complies with the defined quality factors.

In the case of scores that do not naturally appear as percentage scores, a target score can be used to get a percentage. QCF allows managers to drill down and identify weak quality attributes. As a score is computed for each goal, sub goal and attribute individually, these scores can be used to identify weak points in the project that need further improvement.

CONCLUSION

In this paper, we have presented a framework for measuring the quality of web-based systems particularly for maintenance, reliability and usability. The framework we have presented is by no means a final conclusion on how web-based systems can be measured, but we have provided a framework which can be extended by it's users, and we believe that this is a step to more effective measurements of web quality.

We acknowledge that our framework does not include a complete set of objective measurements to cover every aspect of web quality. It is expected that organisations need to define their own quality factors, criteria and metrics specific to their system context. The use of some of our metrics for maintainability becomes more difficult when the site has a lot of dynamic contents. Links that are generated from a database should not be counted the same as static links, but this is difficult to determine on the client side. Over time, we expect that methods can be found to measure objectively many of the criteria we have proposed.

For the future, we suggest creation of an open knowledge base of web-based software's quality factors, criteria, and metrics. Software professionals can then make use of already defined model that suits them or find the closest model and modify it according to their needs.

REFERENCES

[1] Boehm, B., Brown, J., Lipow, M., "Quantative evaluation of software quality", Proc. Int'l conf. on software Engineering, IEEE Computer Society press, 1976.

[2] W3C:XSL Transformations (XSLT), *http://www.w3.org/TR/xslt* (website)

[3] L. Olsina, D. Godoy, G. J. Lafuente and G. Rossi, "Specifying Quality Characteristics and Attributes for Websites", *Web Engineering*, 2001, pp. 266-278.

[4] R. Fewster, E. Mendes, "Measurement, Prediction and Risk Analysis for Web Applications", *7th International Software Metrics Symposium*, 2001, pp 338-348.

[5] D. Reifer, "Web Development: Estimating Quick-to-Market Software", *IEEE Software*, November/December 2001, pp 57-64.

[6] G. M. Weinberg: *The Psychology of Computer Programming*, 1979.

[7] Cavano, J., McCall, J., "A Framework for the Measurement of Software Quality", Proc. ACM Software Quality Assurance Workshop, November 1978, pp. 133-139.

[8] J. Offutt, "Quality Attributes of Web Software Applications", *IEEE Software*, IEEE, March/April 2002, pp. 25-32

[9] Fenton, N. E., Pfleeger, S. L, *Software Metrics a Rigorous and Practical Approach*, PWS Publishing Company, International Edition, 1997

[10] Pressman. R, *Software Engineering A Practitioner's approach*, McGraw-Hill, International Edition, 1997

[11] Ejiogu, L., *Software Engineering with Formal Metrics*, QED Publishing, 1991.