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.

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.

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:

\[
\text{<a\_Quality\_Goal>}
\text{<a\_set\_of\_Criteria\_for\_the\_Goal>}
\text{<a\_set\_of\_Metrics>}
\]

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

\[ L + ER + SC \]

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

\[
\text{Dynamic pages} \times 100
\]

\[
\text{total pages}
\]

- **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:

\[
\text{Number of templates}
\]

**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:

\[
\text{number of format} \times 100
\]

\[
\text{number of pages}
\]

- **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:

\[
\text{number of pages with similar data} \times 100
\]

\[
\text{number of pages}
\]

- **Program/documentation score (PS):** A percentage score averaged 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:

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

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.
Copyright © 2004, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
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

\[ \text{QualityMeasurement} = \sum \frac{\text{Childrens'QCF}}{\text{No. of Children}} \]

- Goal and sub-goal QCF score

\[ \text{Goal QCF score} = \sum \frac{\text{Childrens'QCF}}{\text{No. of Children}} \]

- Attribute QCF score

\[ \text{AttributeQCF} = \frac{\text{EarnedScore}}{\text{PossibleScore}} \times 100\% \]

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.
Related Content

A New Approach to Community Graph Partition Using Graph Mining Techniques
www.irma-international.org/article/a-new-approach-to-community-graph-partition-using-graph-mining-techniques/169175

The Effects of Sampling Methods on Machine Learning Models for Predicting Long-term Length of Stay: A Case Study of Rhode Island Hospitals
www.irma-international.org/article/the-effects-of-sampling-methods-on-machine-learning-models-for-predicting-long-term-length-of-stay/251900

Database Techniques for New Hardware
www.irma-international.org/chapter/database-techniques-for-new-hardware/183909

A Comparative Study of Infomax, Extended Infomax and Multi-User Kurtosis Algorithms for Blind Source Separation
www.irma-international.org/article/a-comparative-study-of-infomax-extended-infomax-and-multi-user-kurtosis-algorithms-for-blind-source-separation/219807

Information Science and Technology in Franchising
www.irma-international.org/chapter/information-science-and-technology-in-franchising/112475