Volume 3, No. 7, July 2012

Journal of Global Research in Computer Science

ISSN-2229-371X

REVIEW ARTICAL

Available Online at www.jgrcs.info

MAINTAINABILITY ASSESSMENT OF WEB BASED APPLICATION

Laxmi Shanker Maurya[1], Gauri Shankar*[2]

[1]Associate Professor, Department of Information Technology,
Shri Ram Murti Smarak College of Engg. & Tech, Bareilly (U.P.), INDIA
lsmaurya@yahoo.com

[2]Shri Ram Murti Smarak College of Engg. & Tech, Bareilly (U.P.), INDIA
gaurishankar021@gmail.com

***Abstract:* -** Many World Wide Web (www) application incorporate important business assets and after a convenient way for business to promote their services through internet. Many web applications have evolved from simple HTML page to complex application [1], some are evolved from different web technologies that have a high maintenances cost. As during evolution of web application well-defined software process and methodologies are used for development of application.

Several maintainability models have been defined for measuring the maintainability of web application. It is very difficult to choose maintainability model. So a strong need for methods and models which have some common criteria to assess the maintainability model of existing web applications.

Keywords—Maintainability Model, Maintainability metrics, web based applications.

INTRODUCTION

In the present scenario, the web based applications are growing faster and exponentially. The main reason behind this is diffusion of internet in the world. Due to market demand web applications are developed frequently and quickly. Many of them incorporate with many different assets like business related asset, education related assets, information related assets, and security related assets and many more. These web application helps in providing information, a convenient way for business, education and sharing information , all these done to promote the services of web applications through internet. As society becomes more and more dependent on software and demands that new, more capable software be provided on short cycles, the need for maintainable, reliable software continues to increase.

Now a day web applications are evolved from simple HTML pages to complex applications which may be evolved from Java Server Pages (JSP), Active Server Pages (ASP), Servlets and many other technology. As the complexity of web Applications are increased the difficulty to maintain these application are also increased. Internet has evolved in terms of number of web sites and number of usage tremendously during the last decade. An example for internet evolution is amazon.com a leading-commerce web application. Amazon.com started with 0customers in 1995. In 2003 it had around 20 million customers and the largest online store in 220 countries [2].Rapid growth of the internet and enormous evolution with very short project release cycles and high competition resulted in many unreliable web applications.

The well-known software engineering principles are not usually applied, as well as well-defined software processes and methodologies are rarely adopted. As a consequence of such an undisciplined development and evolution, a WA usually presents disordered architectures, poor or non-Existing documentation, and can be analyzed, comprehended and modified with a considerable effort [3].

To avoid a software crisis for the WAs there is a strong need to urgently address the definition and the experimentation of methodological approaches, techniques, and tools supporting an effective maintenance of existing web applications. Analogously, there is a strong need also for methods and models to assess the maintainability of existing WAs in order to have a valuable support to successfully estimate the effort of maintenance intervention [3].

While dealing with WA's maintainability assessment, the first step to achieve is the defining of software attributes affecting maintainability; the related model for such WAs will be consequently carried out.

THE MAINTAINABILITY PROBLEM:

The WAs are built under high pressure to meet deadlines, and with initial emphasis on performance, reliability, and usability. The attributes relating to later changes in the software – maintainability attributes are:

a. never specified quantitatively up front in the software quality requirements
b. never architected to meet the non-specified maintainability quality requirements
c. never built to the unspecified architecture to meet the unspecified requirements
d. never tested before release
e. Never measured during the lifetime of the system.

"A number of people expressed the opinion that code is often not designed for change. Thus, while the code meets its operational specification, for maintenance purposes it is poorly designed and documented [4]."

The difficulty in maintaining web applications is due to **Lehman's laws of software evolution**. These laws are as follows [5]:-

a. ***Continuing change* —** An E-type program that is used must be continually adapted else it becomes progressively less satisfactory.

b. ***Increasing complexity* —** As a program is evolved, its complexity increases unless work is done to maintain or reduce it.

c. ***Self-regulation* —** The program evolution process is self-regulating with close to normal distribution of measures of product and process attributes.

d. ***Invariant work rate* —** The average effective global activity rate on an evolving system is invariant over the product lifetime.

e. ***Conservation of familiarity* —** During the active life of an evolving program, the content of successive releases is statistically invariant.

f. ***Continuing growth* —** Functional content of a program must be continually increased to maintain user satisfaction over its lifetime.

g. ***Declining quality* —** E-type programs will be perceived as of declining quality unless rigorously maintained and adapted to a changing operation environment.

h. ***Feedback system* —** E-type programming processes constitute multi-loop, multi-level feedback systems and must be treated as such to be successfully modified or improved.

In addition to this, web applications have some characteristics that make their maintenance costly: heterogeneity speed of evolution, and dynamic code generation. In order to control the maintenance cost of web applications, quantitative metrics for predicting web applications maintainability must be used. Web applications are different from traditional software systems, because they have special features such as hypertext structure, dynamic code generation and heterogeneity that cannot be captured by traditional and object-oriented metrics, hence metrics for traditional systems cannot be applied to web applications [6].

Here now we switch to know some maintainability metrics which affect the maintainability for WA. We take the reference model of ***Oman and Hagemeister maintainability model*** for software. Oman and Hagemeister presented a maintainability model based on a hierarchical tree structure comprehending 92 attributes affecting the Maintainability of a software system. The leaf nodes in the hierarchy represent an identified maintainability attribute and, for each of these, attribute metrics are defined to evaluate that maintainability characteristic. In Figure 1 the top level of the OHMM hierarchy is showed. At this level, three main categories of factors are pointed out:

a) Management: practices of management employed, and facts related with them;
b) Operational environment: environment, in terms of hardware and software, involved in the operation of the system under examination;
c) Target Software System: the examined software system under maintenance, including the source code and support documentation.

Oman's work focuses mainly on the Target Software System; Figure 2, shows a detail of the sub-tree concerning this category.

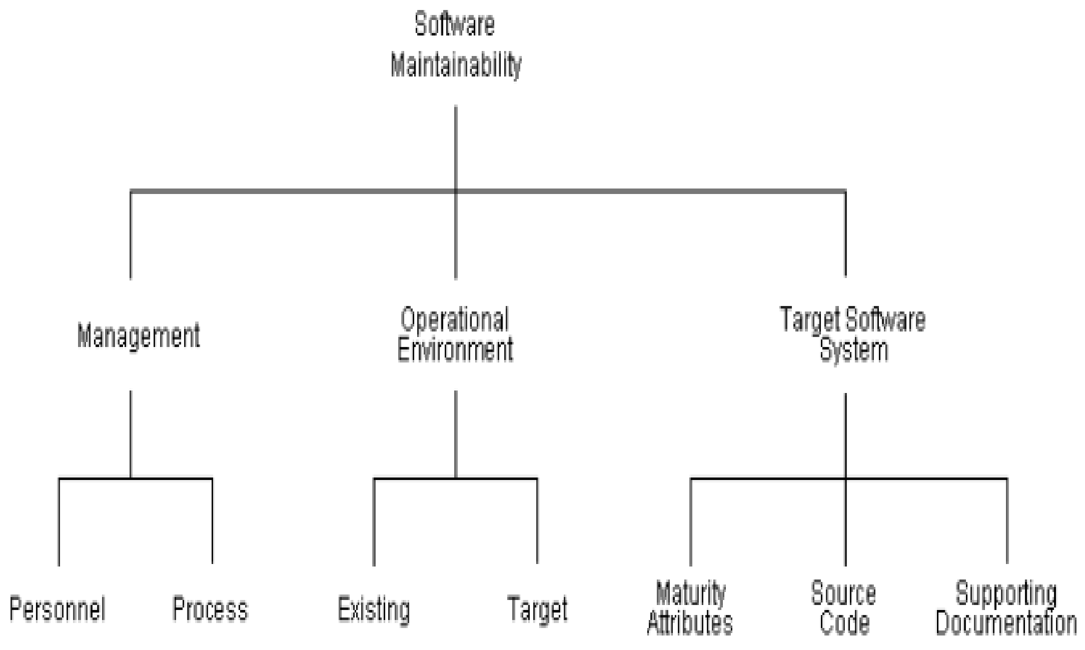

Figure 1. The Oman and Hagemeister top level Software Maintainability hierarchy

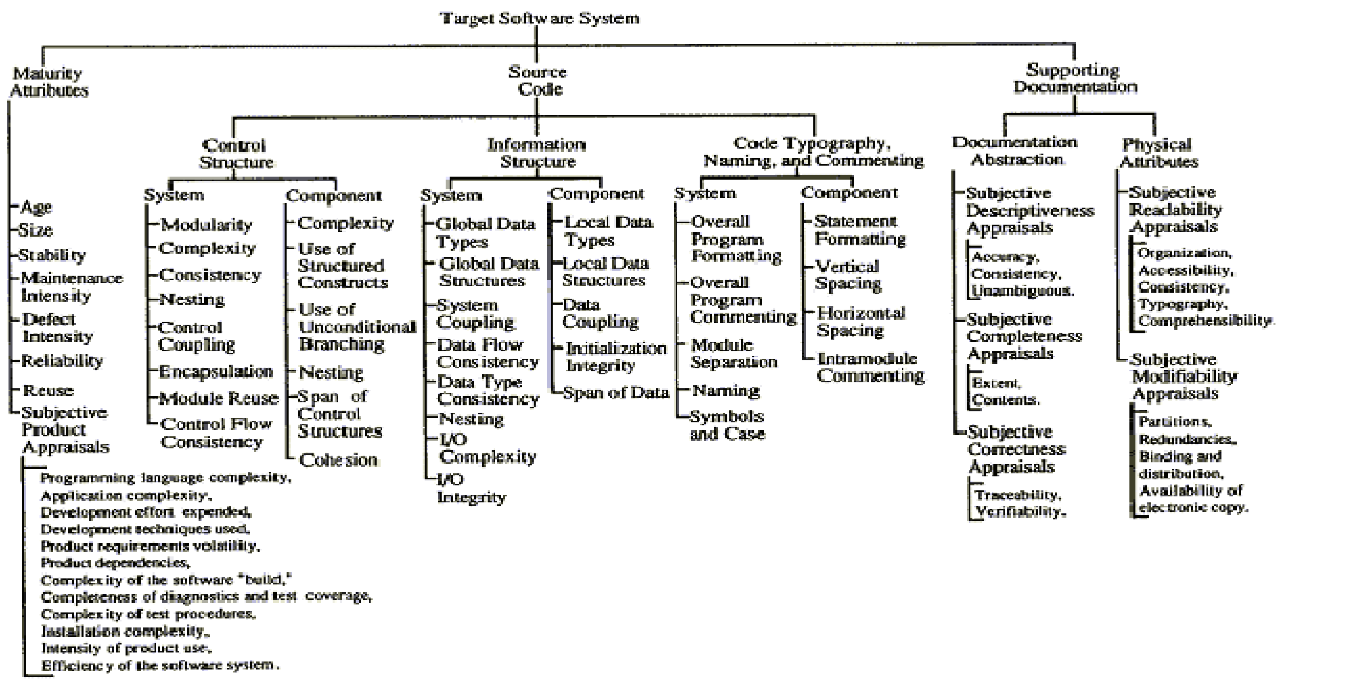
Target Software System
Maturity Attributes
Age
Size
Stability
Maintenance Intensity
Defect Intensity
Reliability
Reuse
Subjective Product Appraisals
Programming language complexity,
Application complexity,
Development effort expended,
Development techniques used,
Product requirements volatility,
Product dependencies,
Complexity of the software "build,"
Completeness of diagnostics and test coverage,
Complexity of test procedures,
Installation complexity,
Intensity of product use,
Efficiency of the software system.
Source Code
Control Structure
System
Modularity
Complexity
Consistency
Nesting
Control Coupling
Encapsulation
Module Reuse
Control Flow Consistency
Component
Complexity
Use of Structured Constructs
Use of Unconditional Branching
Nesting
Span of Control Structures
Cohesion
Information Structure
System
Global Data Types
Global Data Structures
System Coupling
Data Flow Consistency
Data Type Consistency
Nesting
I/O Complexity
I/O Integrity
Component
Local Data Types
Local Data Structures
Data Coupling
Initialization Integrity
Span of Data
Code Typography, Naming, and Commenting
System
Overall Program Formatting
Overall Program Commenting
Module Separation
Naming
Symbols and Case
Component
Statement Formatting
Vertical Spacing
Horizontal Spacing
Intramodule Commenting
Supporting Documentation
Documentation Abstraction
Subjective Descriptiveness Appraisals
Accuracy, Consistency, Unambiguous.
Subjective Completeness Appraisals
Extent, Contents.
Subjective Correctness Appraisals
Traceability, Verifiability.
Physical Attributes
Subjective Readability Appraisals
Organization, Accessibility, Consistency, Typography, Comprehensibility.
Subjective Modifiability Appraisals
Partitions, Redundancies, Binding and distribution, Availability of electronic copy.

Figure 2. The Target Software System subtree of the Oman and Hagemeister Software Maintainability hierarchy

ADAPTING THE OMAN AND HAGEMEISTER MAINTAINABILITY MODEL TO WAS

The differences between traditional systems and Was having to be considered to apply the OHMM to WAs: the original model has to be modified to be efficiently and effectively used with Was.

Firstly, we define a set of simple metrics characterizing a WA either at system and component level, then we will use these metrics to evaluate the attributes. Table 1 and Table 2 report these metrics.

Table 1. WA Metrics at System Level

Metric Name	Description
TotalWebPage#	Total number of WA pages TotalWebPage#= TotalServerPage# + TotalStaticClientPage#
TotalLOC#	Total number of WA LOCs TotalLOC# = TotalServerPageLOC# + TotalStaticClientLOC#
ServerScript#	Total number of WA server scripts
ClientScript#	Total number of WA client scripts
WebObject#	Total number of WA web objects
InterfaceObject#	Total number of WA Interface Object
TotalData#	Total number of different data identifiers
I/OField#	Total number of WA form fields + number of I/O data from/to mass storage devices
TotalConnectivity#	Total number of relationships among web pages TotalConnectivity# = Link# + Redirect# + Submit# + Build#+Include#
TotalLanguages#	Total number of programming/scripting languages used to implement the WA

The metrics in the Table 1 aim to provide a structural size of the whole WA by counting the total number of components it consists of.

The metrics in Table 2 aim to provide information about the structural complexity of a web page both by its internal composition and when connected to other pages.

Table 2. WA Metrics at Component Level

Metric Name	Description
WebPageTag#	Number of tags in the page
WebPageScript#	Number of scripts in the page
PageWeb Object#	Number of web objects in the page
WebPageI/O#	Number of form fields in the page + number of I/O data from/to mass storage devices
WebPageRelationships#	Number of relationships that page has with the other pages WebPageRelationships#= PageLink# + PageRedirect# + PageSubmit# + PageBuild#+PageInclude#
PageCodeSize	Number of source LOCs forming the page
PageInterface#	Number of Interface Objects referred in the page
WebObjectSize	Number of source LOCs forming the web object Note: This metric is used just only for those web objects whose source code is available
WebPageData#	The number of data different data identifiers in the Page
WebPageDataCoupling#	The number of data exchanged with other Web Pages
InnerComponents#	The number of inner components composing the page: InnerComponents# = PageForms# + PageWebObjects# + PageScripts# + PageFrames#
WebPageComplexity#	The cyclomatic complexity of the page
WebPageControlStructures#	The number of Control Flow Structures
PageLanguage#	Number of programming/scripting languages used to implement the page
ScriptSize	Number of source LOCs forming the Script

In the OHMM the main system basic unit is the module or program, that is mainly characterized by its size in KLOC, the data it refers to, the number of control flow structures used to implement it, the control and data coupling it has with the other modules or programs. In a WA the basic unit is the Web Page that is mainly characterized by its inner components (forms, scripts, web objects, and so on - we will refer these components as page sub-components in the following), its size in LOC, the tags and the control flow structures used to implement it, the data it refers, the connections it has with other pages.
These data were taken from the [3].

The Maintainability of a WA, with reference to the Source Code Control and Information Structure characteristics, may be expressed as a function of the 39attributes:
WA Maintainability = F(γ_i,A_i) i=1 .. 39

Where Ai is the value of i^{th} maintainability attribute and γ_i is the weight to assign to that attribute according to how much the attribute affects the maintainability [3].

Thus we have seen from the "Oman and Hagemeister maintainability model (OHMM)", that there are 39 main metrics which affect the web based application or WA.

These are related to the maintainability with a function. This function denotes a closer look to the maintainability of a WA. The WA is not a very short area in evolutions or in testing it is as much difficult to maintain it as in developing it.

WA maintainability is going to be a considerable challenge for many years to come. The systems being maintained are becoming increasingly complex, and a growing proportion of WA development staff is participating in the maintenance of industrial software systems. Although these models are not perfect, they demonstrate the utility of such models. The point is that a good model can help maintainers guide their efforts and provide them with much needed feedback. Before developers can claim that they are building maintainable systems, there must be some way to measure maintainability.

CONCLUSION

Web applications have evolved into complex applications that have high maintenance cost. The high cost is dueto the inherent characteristics of web applications, to the rapid evolution of the Internet, and to the pressing market which imposes short development cycles and frequent medications. In order to control the maintenance cost of web applications, quantitative metrics for predicting web applications maintainability must be used.

When properly interpreted the metrics can provide useful indications about the quality of the WA to be maintained, in terms of: size, coupling, data, and complexity.

The rapid diffusion of different business domains' services over the Web entails critical conditions of development and maintenance for Web Applications. The time-to-market for new WAs ever more shortened and modifications requests for existing was are ever more frequent. Due to both such a market pressure and to the lack of widely spread and validated methodologies, quality of WAs is seriously affected, especially in terms of maintainability.

REFERENCES

[1]. A General Evaluation Criteria for Web Applications Maintainability Models. Emad Ghosheh, Sue Black, Jihad Qaddour- 2008 IEEE.

[2]. L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice. Addison-Wesley, 2 edition, 2003.

[3]. Towards the definition of a maintainability model for web applications- Giuseppe Antonio Di Lucca, Anna Rita Fasolino, Porfirio Tramontana, Corrado Aaron Visaggio- 2004 IEEE

[4]. Susan Dart , Alan M. Christie , Alan W Brown -A Case Study in Software maintenance, Technical Report CMU/SEI-93-TR-8 , ESC-TR-93-185 , June 1993

[5]. M. Lehman, J. Ramil, P. Wernick, D. Perry, and W. Turski. Metrics and laws of software evolution the nineties view. In Proceedings of the 4th International Software Metrics Symposium, pages 20–32. IEEE Computer society Press, 1997.

[6]. Design Metrics forWeb Application Maintainability Measurement- Emad Ghosheh, Sue Black, Jihad Qaddour- 2008 IEEE.

Short Bio Data for the Author

Laxmi Shanker Maurya is an Associate Professor in the Department of Information Technology at Shri Ram Murti Smarak College of Engineering and Technology, Bareilly. He is B. Tech. in Computer Engineering from GBPUAT Pant agar, M. Tech. in Information Technology from AAIDU Allahabad and MBA in HR from IGNOU New Delhi.

Gauri Shankar is persuing M. Techin Software Engineering from Shri Ram Murti Smarak College of Engineering and Technology, Bareilly. He is B.Tech in Computer Science in Engineering KIT, Kanpur (Gautam Buddha Technical University, Lucknow).