A comparative analysis of maintainability approaches for web applications.

Emad Ghosheh[1]
Jihad Qaddour[2]
Matthew Kuofie[2]
Sue Black[3] *

[1] IBM-Global Services, Kansas City, MO 64138, USA
[2] School of Information Technology, Illinois State University
[3] School of Business, Computing and Information Management, London South Bank University

* Sue Black now works within the Harrow School of Computer Science at the University of Westminster

A Comparative Analysis of Maintainability Approaches for Web Applications

Emad Ghosheh , Jihad Qaddour , Matthew Kuofie and Sue Black

Abstract—

Web applications incorporate important business assets and offer a convenient way for businesses to promote their services through the internet. Many of these web applications have evolved from simple HTML pages to complex applications that have high maintenance cost. The high maintenance cost of web applications is due to the inherent characteristics of web applications, to the fast internet evolution and to the pressing market which imposes short development cycles and frequent modifications. In order to control the maintenance cost, quantitative metrics and models for predicting web applications' maintainability must be used. Since, web applications are different from traditional software systems, models and metrics for traditional systems can not be applied to web applications. The reason for that is that web applications have special features such as hypertext structure, dynamic code generation and heterogenousity that can not be captured by traditional and object-oriented metrics. In this paper, we will provide a comparative analysis of the different approaches for predicting web applications' maintainability and point out areas that need further research.

Index Terms—**Maintainability, web applications, regression analysis, WebMo, WAMM.**

I. Introduction

MANY World Wide Web applications incorporate important business assets and offer a convenient way for businesses to promote their services through the Internet. Many of these web applications evolved from simple HTML pages to complex applications which have high maintenance cost. This is due to the laws of software evolution and to some special characteristics of web applications. The following are two software evolution laws [1] that affect the evolution of web applications:

1. The law of continuing change: A program used in real world must change or eventually it will become less useful in the changing world.
2. The law of increasing complexity: As a program evolves it becomes more complex and extra resources are needed to preserve and simplify its structure.

In addition to the reasons above, web applications have some characteristics that make their maintenance costly such as heterogenousity, technology advancement, speed of evolution, dynamic code generation, duplicated code, and tangled and scattered code.

Emad Ghosheh
IBM-Global Services,
Kansas City, MO 64138, USA.
Phone: +1 816 761-8754 email: eghosheh@us.ibm.com
Dr. Jihad Qaddour
School of Information Technology,
Illinois State University, Normal, IL 61790-5150, USA.
Phone: +1 309 438-8146 email: jqaddou@ilstu.edu
Dr. Matthew Kuofie
School of Information Technology,
Illinois State University, Normal, IL 61790-5150, USA.
Phone: +1 309 438-3741 email: mkuofie@ilstu.edu
Dr. Sue Black
School of Business, Computing and Information Management,
London South Bank University, London SE1 0AA, UK.
Phone: +020 7815 7471 email: blackse@lsbu.ac.uk

Web applications are different from traditional software systems: Models and metrics for traditional systems can not be applied to web applications. The reason for that is that web applications have special features such as hypertext structure, dynamic code generation and heterogenousity that can not be captured by traditional and object-oriented metrics. Another difference is the difference in the unit of measurement of a metric for each application domain. For traditional systems, the unit of measurement of a metric can be a file, procedure or an attribute. For object-oriented systems, the unit of measurement can be a class, interface or attributes. For web applications, the unit of measurement is a web object which can be either an HTML file, JSP, Servlet or client scripts.

The remainder of this paper is organized as follows: Section 2 provides a review on related research. Section 3 provides a comparative analysis of the different approaches for predicting web applications' maintainability. Finally, section 4 provides a conclusion and describes future wok to be undertaken.

II. Related Work

The cost of software maintenance accounts for a large portion of the overall cost of a software system [2], [3]. Software maintenance can be categorized into perfective, adaptive, corrective and preventive maintenance.

Perfective maintenance improves the functionality of the software system by expanding requirements. Adaptive maintenance deals with porting a software system to a new hardware or software environment. Corrective maintenance deals with modifications associated with errors in the software system. Preventive maintenance deals with software modifications that prevent possible future errors. One of the main concerns of system stakeholders is to increase the maintainability of the software system. Maintainability can be defined as the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment [2], [4]. Maintainability can be measured by measuring some of the sub-characteristics of maintainability such as understandability, analyzability, modifiability and testability. Some studies have measured maintainability by measuring both modifiability and understandability [5], [6], [7]. Understandability is an important sub-characteristics of maintainability, since professionals spend at least half of their time analyzing software to understand it [8]. In some studies the maintainability has been quantified in the Maintainability Index (MI) [9], [10], [11]. The Maintainability Index is measured as a function of directly measurable attributes A_1 through A_n as shown in Equation 1:

$$M = f(A_1 + A_2 + ... + A_n) \quad (1)$$

The measure (M) is called a Maintainability Index which can

differ depending on the attributes being used in the measurement. For example, Sneed [12] proposed a software quality assessment environment called SOFTING. SOFTING uses the following design attributes: modularity, portability, integrity, redundancy, complexity, generality, time and span-utilization with associated metrics to calculate the maintainability index. Other studies used the effort for measuring maintainability [13]. In [14] change effort, which is the total analysis and programming effort spent on a particular change is used to measure maintainability. There has been a lot of effort in constructing formulas for measuring maintainability. The following are the different approaches used to quantify maintainability from software metrics [9]:

- Hierarchical Multidimensional Assessment: In this technique the attributes are defined in a hierarchy. The top level is divided into three levels control structure, information structure and documentation. Each level is assigned to certain metrics. The total maintainability index is calculated by adding up all the metrics in the hierarchy [4].
- Aggregate Complexity Measure: In this technique the maintainability is calculated using a function of entropy [15].
- Regression Analysis Models: In this technique a polynomial equation is constructed to measure maintainability using a function of metrics [4].
- Factor Analysis: This is a statistical technique where metrics are grouped into clusters where each cluster has metrics that are highly correlated to each other and lowly correlated to metrics in other clusters. Each group of metrics presents a single underlying factor [15].
- Principal Components Analysis: This is also a statistical technique that reduces the collinearity between independent variables. It will reduce the number of independent variables used to construct a maintainability regression model [16].

All these models were tested and validated on Hewlett-Packard systems. They showed accurate measures of maintainability from simple metrics. Out of the five two were named as being the most practical ones namely, Regression Analysis and Hierarchical Multidimensional Assessment. Most of the studies related to maintainability measurements have been on structured and object-oriented systems. Little work has been done with this regard on web applications.

III. Comparison and Analysis

Most of the studies related to maintainability measurements have been on structured and object-oriented systems. Little work has been done with this regard on web applications. Most of the studies applied on web application measure maintainability using the effort or the Maintainability Index. None of the studies has used understandability, analyzability, modifiability and testability in measuring maintainability. Most studies used source code metrics for measuring maintainability. The drawback of using source code metrics is that prediction can only be made later in the development project. Early measurements are much better since they help in mitigating risks early in the development process.

Table I shows a comparison between the different approaches used for predicting maintainability of web applications. The first approach used the Oman and Hagemeister Hierarchical Tree model. WAMM [11] used source code metrics and the maintainability was measured using the Maintainability Index. In WAMM new metrics were defined but there is still a need to validate those metrics empirically and theoretically. There is a need to prove how practical WAMM will be in an industrial environment since WAMM captures a lot of metrics which might make it unpractical to implement unless there is a tool that can capture all the metrics and provide a single Maintainability Index with a click on a button. The most common approach used is the Regression Analysis approach. There a couple of studies in the literature using Regression Analysis to define and validate metrics and models for web applications. In [17] design and authoring effort were the dependent variables. The independent variables were based on source code metrics. There is still a need to do more empirical studies to validate the new defined metrics in order to make general conclusions. In [18] design metrics were introduced based on W2000 which is a UML like language. In the study the dependent variables were variations of design effort. The independent variables were measured from the presentation, navigational and information models. Some data for the presentation model was discarded in the study due to lack of participation from all subjects. It is not known how useful this approach would be, since it is not known if the W2000 language is used outside the educational environment and if it will become popular in industrial environments. In [19] Maintenance Time is used as the dependent variable and some metrics based on the Navigational model are used as independent variables. In this approach like the previous one it is not known how practical this approach is. WebMo [20] introduces the notion of Web Objects as size measures for predicting the effort of developing web applications. There is a need for more empirical studies to prove the benefit of WebMo. Case Based Reasoning [21], [22] is an approach that uses a number of projects' features stored in a database to predict the effort of the current project. The main problem with this approach is that web applications keep changing and new technologies are introduced all the time. Therefore, there is a need to always update the features database by adding new projects and removing obselete projects that use old technologies. This might become unpractical unless there is a way to automate that process.

IV. Conclusion

Web applications are one of the fastest growing classes of software systems. They have diffused in many and different business domains such as scientific activities, product sale and distribution and medical activities[23], [24]. These web applications have evolved into complex applications that have high maintenance cost. The high maintenance cost of web applications is due to the inherent characteristics of web applications, to the fast Internet evolution and to the pressing market which imposes short development cycles[24] and frequent modifications. An example for Internet evolution is amazon.com a leading e-commerce web application. Amazon.com started with 0 customers in 1995. In 2003 it had around 20 million customers and the largest online store in 220 countries[25]. In order to control the maintenance cost of web applications, quantitative

Study	Dependent Variable	Independent Variable	Validation	Comments
WAMM [11]	Maintainability Index	Oman and Hagemeister source tree (Control Structure and Information Structure at System and Component level)	Empirical Validation	Little Empirical Validation, No Theoritical Validation
Regession Analysis [17]	Design and Authoring Effort	Source Code Metrics (PageCount, MediaCount, ProgramCount, TotalPageAllocation, TotalMediaAllocation, TotalEmbeddedCodeLength, ReusedMediaCount, ReusedProgramCount, TotalReusedMediaAllocation, TotalReusedCodeLength, Connectivity, ConnectivityDensity, TotalPageComplexity, CyclomaticComplexity, SizeCFSU, Structure)	Empirical Validation	Little Empirical Validation, No Theoritical Validation
Regession Analysis [18]	Information Effort, Navigation Effort, Presentation Effort, and Total Design Effort	W2000 UML like language (design metrics for Navigational Model, Informational Model, Presentation Model)	Empirical Validation	Little Empirical Validation, No Theoritical Validation, Metrics for Presentation Model were excluded since it was not mandatory in the experiment.
Regession Analysis [19]	Maintenance Time	Navigational Model Metrics (Number of Navigational Contexts (NNC), Number of Navigational Links (NNL), Density of a Navigational Map (DeNM), Depth of a Navigational Map (DNM), Breadth of a Navigational Map (BNM), Minimum Path Between Navigational Contexts (MPBNC), Number of Path Between Navigational Contexts (NPBNC), Compactness (Cp))	Empirical Validation and Theoritical Validation	Little Empirical Validation
WebMo [20]	Effort and Duration	Size Metrics called Web Objects (Computed using Halsteads equation for volume)	Empirical Validation	Little Empirical Validation, No Theoritical Validation
Case Based Reasoning [21], [22]	Design and Authoring Effort	Project Features (Total Web Pages, Total Images, Total Animations, Total Effort)	Empirical Validation	The number of projects used is still small, Need more projects to generalize results, No Theoritical Validation

TABLE I
A COMPARISON OF WEB MAINTAINABILITY APPROACHES

metrics and models for predicting web applications' maintainability must be used. The maintainability metrics and models can be useful for:

- Predicting the maintenance cost to provide accurate estimates in a project lifecycle[13].
- Comparing different design documents to select the documents that have the highest maintainability.
- Identifying risky components to mitigate risks early in the project by reengineering or allocating experienced developers to the risky components[26].
- Improving the software development process[27].

In this paper we have provided an introduction to the high maintenance cost problem for web applications. We discussed related research in the area of maintainability models. Towards the end of the paper we provided a comparative analysis of different approaches for predicting maintainability of web applications and pointed out areas that were lacking and needed further research.

A. Future Work

As a conclusion, there are many avenues of research in building metrics and models for predicting web applications' maintainability. Most of the studies are still exploratory and need

further validation. Some metrics have been defined for web applications. But still, there is a need to provide theoretical and empirical validation for these metrics so that they can be accepted in the software community. In our future work we will build a maintainability prediction model for web applications using early design metrics based on Web Application Extension (WAE) [28]. WAE is a modeling language that has extensions for UML to capture the special features of web applications. We will use the following design attributes as a basis for defining the metrics for WAE: Coupling [5], Cohesion [5], Clarity [5] Complexity and Size [13], Simplicity [5] and Reusability. We will measure two important components of design maintainability for web applications namely: understandability and modifiability. These two components have been used for developing maintainability models for object oriented systems[5] and still have not been used in the context of web applications. To summarize our future work will include the following:

- Provide a maintainability prediction model for web applications based on statistical regression analysis. This model will provide prediction measures for different maintainability measures, such as Understandability Time, Modifiability Time, Understandability Correctness, Modifiability Correctness and Modifiability Rate.
- Provide a validation of a number of product design metrics for web applications. This validation includes the statistical significance and magnitude that the design product metric has on maintainability measures. The validation of the product metrics will be done using correlation and regression analysis.
- Provide a complete set of empirical experiments for web applications that can be generalized to other settings. These experiments will build on previous research and provide a starting point for further research on web application's maintainability.
- Provide a comparison between our prediction model and other models.
- Provide an environment for the maintainability prediction model that includes tools and procedures so that it can be used in an industrial environment.

REFERENCES

[1] 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*. IEEE Computer Society Press, 1997, pp. 20–32.

[2] Pankaj Bhatt, Gautam Shroff, and Arun Misra, "Dynamics of software maintenance," *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 4, pp. 1–5, 2004.

[3] Norman Wilde and Ross Huitt, "Maintenance support for object-oriented programs," *IEEE Transactions on Software Engineering*, vol. 18, no. 12, pp. 1038–1044, 1992.

[4] Paul Oman and Jack Hagemeister, "Construction of testing polynomials predicting software maintainability," *Journal of Software Systems*, vol. 27, no. 3, pp. 251–266, 1994.

[5] Lionel Briand, Christian Bunse, and Jong Daly, "A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs," *IEEE Transactions on Software Engineering*, vol. 27, no. 06, pp. 513–530, 2001.

[6] Marcela Mario, Esperanza Manso, and Giovanni Cantone, "Building uml class diagram maintainability prediction models based on early metrics," in *Proceedings of the 9th International Software Metrics Symposium*. IEEE Computer Society Press, 2003, pp. 263–278.

[7] Matinee Kiewkanya, Nongyao Jindasawat, and Pornsiri Muenchaisri, "A methodology for constructing maintainability model of object-oriented design," in *Proceedings of the 4th International Conference on Quality Software*. IEEE Computer Society Press, 2004, pp. 206–213.

[8] Thomas Corbi, "Program understanding: challenge for the 1990s," *IBM Systems Journal*, vol. 28, no. 2, pp. 294–306, 1989.

[9] Don Coleman, Dan Ash, Bruce Lowther, and Paul Oman, "Using metrics to evaluate software system maintainability," *IEEE Computer*, vol. 27, no. 8, pp. 44–49, 1994.

[10] Welker Kurt and Paul Oman, "Software maintainability metrics models in practice," *Crosstalk, Journal of Defense Software Engineering*, vol. 8, no. 11, pp. 19–23, 1995.

[11] Giuseppe DiLucca, Anna Fasolino, Porfirio Tramontana, and Corrado Visaggio, "Towards the definition of a maintainability model for web applications," in *Proceeding of the 8th European Conference on Software Maintenance and Reengineering*. IEEE Computer Society Press, 2004, pp. 279–287.

[12] Harry Sneed and Andrs Merey, "Automated software quality assurance," *IEEE Transactions on Software Engineering*, vol. 11, no. 9, pp. 909–916, 1985.

[13] Fabrizio Fioravanti and Paolo Nesi, "Estimation and prediction metrics for adaptive maintenance effort of object-oriented systems," *IEEE Transactions on Software Engineering*, vol. 27, no. 12, pp. 1062–1084, 2001.

[14] Wiebe Hordijk and Roel Wieringa, "Surveying the factors that influence maintainability," in *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*. ACM Press, 2005, pp. 385–388.

[15] John Munson and Taghi Khoshgoftaat, "The detection of fault-prone programs," *IEEE Transactions on Software Engineering*, vol. 18, no. 5, pp. 423–433, 1992.

[16] Fang Zhuo, Bruce Lowther, Paul Oman, and Jack Hagemeister, "Constructing and testing software maintainability assessment models," in *Proceedings of the 1st International Software Metrics Symposium*. IEEE Computer Society Press, 1993, pp. 61–70.

[17] Emilia Mendes, Nile Mosley, and Steve Counsell, "Web metrics - estimating design and authoring effort," *IEEE Multimedia*, vol. 08, no. 01, pp. 50–57, 2001.

[18] Luciano Baresi, Sandro Morasca, and Paolo Paolini, "Estimating the design effort of web applications," in *Proceedings of the 9th International Software Metrics Symposium*. IEEE Computer Society Press, 2003, pp. 62–72.

[19] Silvia Abraho, Nelly Condori-Fernndez1, Luis Olsina, , and Oscar Pastor, "Defining and validating metrics for navigational models," in *Proceedings of the 9th International Software Metrics Symposium*. IEEE Computer Society Press, 2003, pp. 200–210.

[20] Donald Reifer, "Web development: estimating quick-time-to-market," *IEEE Software*, vol. 17, no. 8, pp. 57–64, 2000.

[21] Emilia Mendes, Nile Mosley, and Steve Counsell, "Early web size measures and effort prediction for web costimation," in *Proceedings of the 9th International Software Metrics Symposium*. IEEE Computer Society Press, 2003, pp. 18–39.

[22] Emilia Mendes, Nile Mosley, and Steve Counsell, "A comparison of development effort estimation techniques for web hypermedia applications," pp. 131–140, 2002.

[23] Sebastian Elbaum, Gregg Rothermel, Srikanth Karre, and Marc Fisher, "Leveraging user-session data to support web application testing," *IEEE Transactions on Software Engineering*, vol. 31, no. 3, pp. 187–202, 2005.

[24] Fillipo Ricca, "Analysis, testing and re-structuring of web applications," in *Proceedings of the 20th IEEE International Conference on Software Maintenance*. IEEE Computer Society Press, 2004, pp. 474–478.

[25] Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice*, Addison-Wesley, 2 edition, 2003.

[26] Khaled EL-Emam, "A methodology for validating software product metrics," Tech. Rep. NRC 44142, National Research Council Canada, 2000.

[27] Shahid Bhatti, "Why quality? iso 9126 software quality metrics (functionality) support by uml suite," *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 2, pp. 1–5, 2005.

[28] Jim Conallen, *Building Web Applications with UML*, Addison-Wesley, 2 edition, 2003.