Design metrics for web application maintainability measurement.

Emad Ghosheh[1]
Sue Black[2]
Jihad Qaddour [3]

[1] Department of Electrical and Electronic Engineering, Eastern Mediterranean University
[2] School of Electronics and Computer Science, University of Westminster
[3] School of Information Technology, Illinois State University

Design Metrics for Web Application Maintainability Measurement

Emad Ghosheh
AT& T, One Bell Center
Saint Louis, MO, USA.
eg2534@att.com

Sue Black
University of Westminster
Department of Information and Software Systems,
Harrow School of Computer Science
London HA1 3TP, UK
s.e.black@westminster.ac.uk

Jihad Qaddour
Illinois State University
School of Information Technology,
Normal, IL 61790-5150, USA
jqaddou@ilstu.edu

Abstract

Many web applications have evolved from simple HTML pages to complex applications that have a high maintenance cost. This high maintenance cost is due to the heterogeneity of web applications, to fast Internet evolution and the fast-moving market which imposes short development cycles and frequent modifications. In order to control the maintenance cost, quantitative metrics for predicting web applications maintainability must be used. This paper provides an exploratory study for new design metrics used for measuring the maintainability of web applications from class diagrams. The metrics are based on Web Application Extension (WAE) for UML and will measure the following design attributes: size, complexity, coupling and reusability. In this study the metrics are applied to two web applications from the telecommunications domain.
Keywords: *Web applications, metrics, maintainability, UML.*

1 Introduction

Many World Wide Web applications incorporate important business assets and offer a convenient way for businesses to promote their services through the Internet. A large proportion of these web applications have evolved from simple HTML pages to complex applications which have high maintenance cost. This is due to the laws of software evolution [11] and to some special characteristics of web applications. Two software evolution laws [11] that affect the evolution of web applications are:

- First Law-Continuing change: a program used in the real world must change or eventually it will become less useful in the changing world.
- Second Law-Growing complexity: as a program evolves it becomes more complex and extra resources are needed to preserve and simplify its structure.

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 can not be captured by traditional and object-oriented metrics, hence metrics for traditional systems can not be applied to web applications.

This paper provides an exploratory study for new design metrics used to measure the maintainability of web applications from class diagrams. The metrics are based on the Web Application Extension (WAE) for UML and measure the following design attributes: size, complexity, coupling and reusability. The remainder of this paper is organized as follows: Section 2 gives a review of web application modeling using UML and discusses related research. Section 3 introduces the new design metrics used for measuring the maintainability of web applications. Section 4 applies the design metrics to two case studies from the telecommuni-

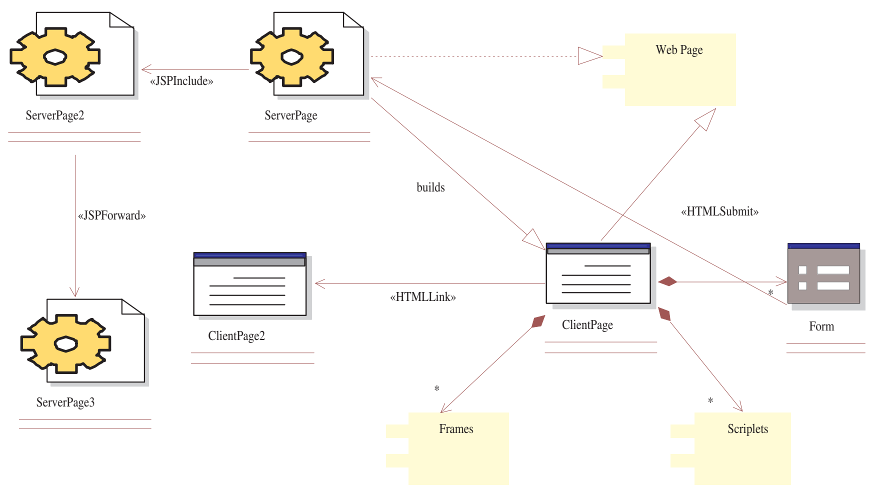

Figure 1. Web Applications Model

cation domain. Finally, section 5 provides a conclusion and describes future work to be undertaken.

2 Background & Related Work

2.1 Web Application Modeling using the Unified Modeling Language(UML)

Modeling is a technique used to represent complex systems at different levels of abstraction, and helps in managing complexity. UML is an object-oriented language [5] that can be used to model object-oriented systems. Web applications are not inherently object-oriented, therefore, it is difficult to use UML to model them, but UML has now been enhanced with extensions to capture web applications various elements. Conallen proposed an extension of UML for web applications [5], Figure 1 shows the elements of the Conallen web application model.

The important elements of Conallen's model are as follows [5]:

- *Web Page*: a web page is the primary element of a web application. It is modeled with two separate stereotyped classes, the client page and the server page. The client page contains client side scripts and user interface formatting. The server page contains server methods and page scoped variables.

- *Relationships*: the model defines the following relations between different components: builds, redirects, links, submit, includes, and forwards. The builds relationship is a directional relationship from the server page to the client page. It shows the HTML output coming from the server page. The redirects relationship is a directional relationship that requests a resource from another resource. The links relationship is an association between client pages and server or client pages. It models the anchor element in HTML. The links relationship can have parameters which are modeled as attributes in the relationship. The submit relationship is a relationship between the form and the server page that processes it. The include relationship is a directional association between a server page and another client or server page. The forward relationship is a directional relationship between a server page and a client or server page. This presents delegating the server request to another page.

- *Forms*: forms are defined to separate the form processing from the client page. The form element contains field elements. Forms are contained in client pages. Each form submits to a different action page.

- *Components*: components run on the client or server page. ActiveX controls and Applets are examples of components.

- *Scriplet*: a scriplet contains references to components and controls that are re-used by client pages.

- *Framesets*: a frameset divides the the user interface into multiple views each containing one web page. Frames can contain more than one client page, but they must contain at least one client page.

- *XML*: an XML element is a hierarchical data representation that can be passed back and forth between client and server pages.

Metric Type	Description
Size	Total number of server pages (NServerP)
	Total number of client pages (NClientP)
	Total number of web pages (NWebP)=(NServerP + NClientP)
	Total number of form pages (NFormP)
	Total number of form elements (NFormE)
	Total number of client components (style sheet and JavaScript components)(NClientC)
Structural Complexity	Total number of link relationships (NLinkR)
	Total number of Submit relationships (NSubmitR)
	Total number of builds relationships (NbuildsR)
	Total number of forward relationships(NForwardR)
	Total number of include relationships(NIncludeR)
	Total number of use tag relationships(NUseTagR)
Control Coupling	Number of relationships over number of web pages: WebControlCoupling = (NLinkR + NSubmitR + NbuildsR + NForwardR + NIncludeR + NUseTagR)/ NWebP)
Data Coupling	Number of data exchanged over number of server pages: WebDataCoupling = (NFormE / NServerP)
Reusability	Number of include relationships over number of web pages: WebReusability = (NIncludeR / NWebP)

Table 1. Web Application Design Metrics

2.2 Related Work

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 [3].

Maintainability can be measured by measuring some of the sub-characteristics of maintainability such as understandability, analyzability, modifiability and testability. Kiewkanya *et al* [10] measured maintainability by measuring both modifiability and understandability. In Coleman *et al* [4] quantify maintainability via their Maintainability Index. The Maintainability Index is measured as a function of directly measurable attributes A_1 through A_n as shown in Equation 1:

$$M = f(A_1, A_2, ..., A_n) \qquad (1)$$

The measure (M) is called a Maintainability Index which can differ depending on the attributes being used in the measurement, Fioravanti *et al* [8] used effort for measuring maintainability.

Most of the studies related to maintainability measurements have looked at structured and object-oriented systems. Little work has been done in this regard with web applications. The Web Application Maintainability Model (WAMM) [6] used source code metrics measuring the maintainability using the Maintainability Index. In WAMM new metrics were defined, but not validated empirically or theoretically. There is also a need to prove how practical WAMM will be in an industrial environment. Two studies use regression analysis to define and validate metrics and models for web applications: in [13] design and authoring effort were the dependent variables. The independent variables were based on source code metrics. There is still a need for more empirical studies to validate these newly defined metrics in order to make general conclusions. In [2] 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 [1] maintenance time is used as the dependent variable and some metrics based on the navigation model are used as independent variables. WebMo [15] introduces the notion of Web Objects as size measures for predicting the effort of developing web applications. Case Based Reasoning [14] is an approach that uses a number of projects features stored in a database to predict the effort of the current project. Further details of web application maintainability approaches are given in [9].

Metric Type	Metric Name	Measurement
Size	(NServerP)	4
Size	(NClientP)	3
Size	(NWebP)	7
Size	(NFormP)	1
Size	(NFormE)	3
Size	(NClientC)	0
Structural Complexity	(NLinkR)	6
Structural Complexity	(NSubmitR)	1
Structural Complexity	(NBuildsR)	3
Structural Complexity	(NForwardR)	1
Structural Complexity	(NIncludeR)	2
Structural Complexity	(NUseTagR)	0
Control Coupling	(WebControlCoupling)	1.86
Data Coupling	(WebDataCoupling)	0.75
Reusability	(WebReusability)	0.28

Table 2. Measurements of Sample Class Diagram

3 UML Web Design Metrics

There are several design quality attributes defined in the literature that have an effect on maintainability such as coupling, cohesion and complexity. This study defines metrics for the following design attributes: size, complexity, coupling and reusability. The metrics are shown in Table 1, and are based on UML class diagrams for web applications. Figure 2 shows a sample class diagram and Table 2 shows the results of calculating the metrics from the sample class diagram. A description of the design attributes is given as follows:

- Size: size can be measured by counting Lines of Code [8]. In this approach the size of a UML class diagram is measured by counting the number of components in the diagram.
- Complexity: complexity can be measured by calculating the number of loops and branches in a component. McCabe Cyclomatic Complexity [12] is a common measure for complexity. This study uses the number of associations and relations in UML class diagrams to measure complexity.
- Coupling: coupling is the degree of interaction between two components [16]. This study measures control and data coupling by analyzing the relationships and data passed between different web components.
- Reusability: reusability is taking components of one product in order to facilitate the development of a different product with different functionality. The study measures the reusability by looking at the percentage of web components that are reused in the whole application.

Characteristic	WebApp1	WebApp2
Application Domain	Telecom	Telecom
Age (years)	4	5
Lines Of Code (LOC)	20K	15K
Language	Java	Java
Web Server	WebLogic 7.0	Tomcat 4.1
Application Server	WebLogic 7.0	None
Framework	Struts 1.1	None
Database	Oracle	MySql
Configuration Management Tool	CVS	None
Design Tool	Rational Rose	None

Table 3. Characteristics of Web Applications

4 Case Study

The proposal of new metrics is not helpful if their practical use is not proved through case studies [7]. This study demonstrates the approach using two web applications case studies. The case studies are exploratory in nature and provide a basis for future research. Both web applications described are from the telecommunication Operational Support System (OSS) domain. Table 3 shows the characteristics of both web applications.

4.1 Case Study Context

The first web application is a provisioning application which is used to provision and activate the wireless service in the network. This study refers to the first web application as WebApp1. WebApp1 has around 10,000 users of which 2,500 are concurrent. It is a critical application that is used by customer care advocates to resolve provisioning issues for wireless subscribers. WebApp1 is built using the latest web technologies and frameworks such Struts, and EJBs and uses Oracle for the database.

In order to further evaluate the metrics this methodology is applied to a second web application. This study refers to the second web application as WebApp2. WebApp2 is a fault management application that is used to automate the configuration management for fault management applications. It is used to automate password resets, creation of

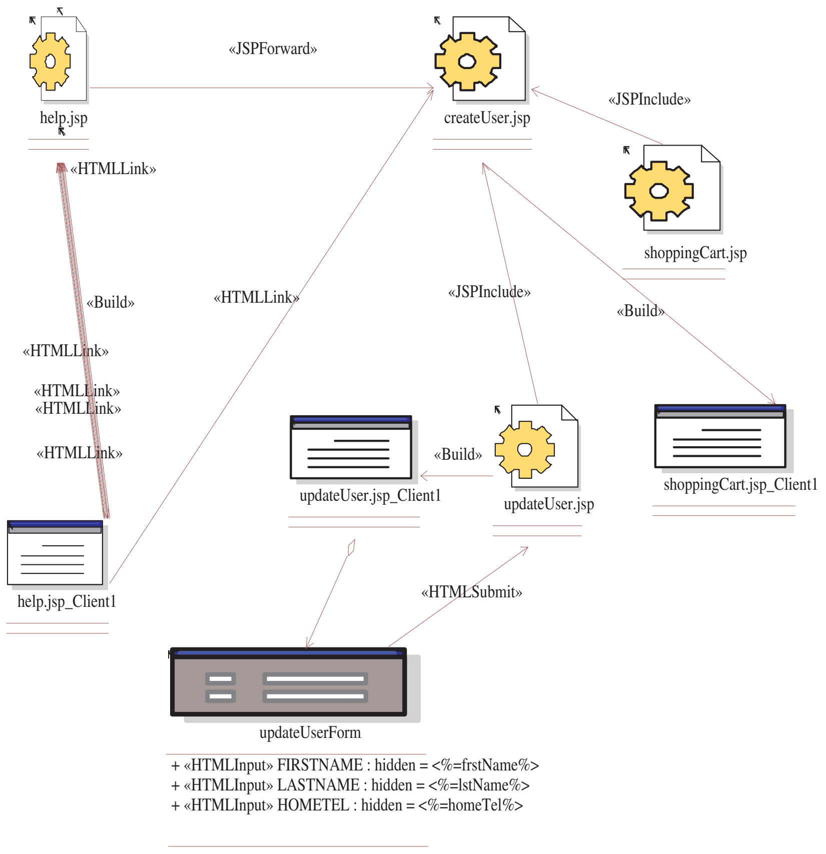

Figure 2. Sample Class Diagram

new users, and creation of configuration management tickets. WebApp2 uses basic web application technologies such as Javascript, servlets, and mysql database.

Both web applications are considered to be of medium size. Due to space limitations the case studies cannot be presented in detail. This study will analyze the web applications based on system metrics.

4.2 Hypothesis

This study aims to answer the following question: Is there a relationship between the metrics identified in Table 1 and maintainability? Since the study is explorative in nature, it measures maintainability in a subjective manner. The maintainability is measured by getting input from the developers on the modifiability maintainability subcharacteristic. The modifiability is based on how easy it is to make changes to the web application.

The following hypotheses are investigated: H1: the lower the size metrics of the class diagram, the higher the modifiability. H2: the lower the structural complexity metrics of the class diagram, the higher the modifiability. H3: the lower the coupling metrics of the class diagram, the higher the modifiability. H4: the lower the reusability metrics of the class diagram, the lower the modifiability.

4.3 Data Collection

In this case study an IBM tool: Rational XDE is used to reverse engineer these web applications. Rational XDE combines the visual modeling with a Java forward and reverse engineering tool. The metrics are measured directly from generated class diagrams. This can become cumbersome if the application is large since a tool was not available to calculate these metrics. In the future we plan to build a tool that can compute these metrics from class diagrams.

In this study several attributes of web applications that are expected to affect maintainability are considered. These attributes include size, complexity, coupling, and reusability. Table 1 gives a description of the metrics that were used in the case studies. This study provided two levels of modifiability measures: low and high. The team lead of each web application was asked to provide a measure for the modifiability based on these two levels.

Metric Type	Metric Name	WebApp1	WebApp2
Size	(NServerP)	35	9
Size	(NClientP)	37	74
Size	(NWebP)	72	83
Size	(NFormP)	11	10
Size	(NFormE)	20	117
Size	(NClientC)	22	158
Structural Complexity	(NLinkR)	44	468
Structural Complexity	(NSubmitR)	8	9
Structural Complexity	(NBuildsR)	35	9
Structural Complexity	(NForwardR)	0	0
Structural Complexity	(NIncludeR)	39	0
Structural Complexity	(NUseTagR)	71	0
Control Coupling	(WebControlCoupling)	2.73	5.85
Data Coupling	(WebDataCoupling)	0.57	13
Reusability	(WebReusability)	0.54	0
Maintainability Measurement	Modifiability	high	low

Table 4. Results

4.4 Results and Analysis

The results are recorded in Table 4. The analysis is as follows:

Looking at the size attribute for WebApp1 one may notice that this application has more server side pages than WebApp2. As a conclusion WebApp2 needs developers with server side development experience, while WebApp1 needs developers with client side development experience. When comparing the structural complexity of WebApp1 to WebApp2, one can notice that WebApp2 has ten times more link relationships than WebApp1 even though the number of web pages for both application is almost the same. WebApp1 and WebApp2 have similar number of submit relationships, but WebApp1 has more form elements per form page, which means it will pass more data for each submit relationship. WebApp1 uses quite a few tags and has many include relationships. On the other hand WebApp2 does not use tags and has no include relationships. While looking at the control coupling for both web applications, one can see that WebApp1 and WebApp2 both have high control coupling due to the number of control relationships in both applications. The data coupling for WebApp2 is very high due to the high number of data elements that are passed between components in the web application. If one looks at reusability one can notice that WebApp1 has good reusability due to the number of include relationships. On the other hand WebApp2 does not demonstrate good reusability that makes maintenance more difficult and makes WebApp2 more prone to error propagation.

WebApp2 has higher size metrics, structural complexity metrics, coupling metrics and lower reusability metrics than WebApp1. It is expected to have lower modifiability. According to the results in Table 4 the modifiability for WebApp1 is high, while the modifiability for WebApp2 is low. Thus we can accept hypothesis H1, H2, H3, and H4.

As a conclusion, this study can serve as a basis for future studies, and can provide a first indication of the use of the newly introduced metrics.

4.5 Threats to Validity

It is important to look at the internal and external validity of this study. In terms of internal validity, firstly, there was no automated tool for collecting the metrics from the design artifacts. There can be some human error in the process of computing the metrics from the class diagrams. Secondly, there was no configuration management tool for WebApp2. As a result some of the components might be out of synch and not representative of the actual application. Also the maintainability is measured in a subjective manner which is less accurate than objective measures. With regard to external validity, one can see that the results can be generalized to other settings. The web applications used are from the telecommunication domain, but they are still using technologies that are similar to other applications in the market. The design of the application was collected using Rational XDE, which requires a license and might not be available for everyone to conduct a similar study, but still the outcome design is based on UML and there are many freeware tools in the market that can be used to generate the design.

5 Conclusions and Future Work

Web applications have evolved into complex applications that have high maintenance cost. The high cost is due to 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 modifications. In order to control the maintenance cost of web applications, quantitative metrics for predicting web applications maintainability must be used.

The maintainability of web applications is a new research area that is becoming important and interesting for researchers in software engineering. Most research in this area is still exploratory and needs further validation. Some metrics have been defined for web applications, but there is still a need to provide theoretical and empirical validation for these metrics so that they can be accepted in the software community. This study has introduced metrics for the following design attributes: coupling, complexity and size, and reusability. This study has provided two explorative case studies from the telecommunication domain utilizing the metrics.

The results give a first indication of the usefulness of the UML design metrics. There is a relationship between maintainability and the metrics. However, the amount and strength of the relationship cannot be determined without further empirical studies. One drawback of the study is the subjective measurement of maintainability, in the future we will measure maintainability objectively and provide statistical analysis to determine the strength of the relationship between the metrics and the measured maintainability. We will provide a maintainability prediction model for web applications based on statistical regression analysis which will provide prediction measures for different maintainability measures.

References

[1] S. Abraho, N. Condori-Fernndez, L. Olsina, and O. Pastor. Defining and validating metrics for navigational models. In *Proceedings of the 9th International Software Metrics Symposium*, pages 200–210. IEEE Computer Society Press, 2003.

[2] L. Baresi, S. Morasca, and P. Paolini. Estimating the design effort of web applications. In *Proceedings of the 9th International Software Metrics Symposium*, pages 62–72. IEEE Computer Society Press, 2003.

[3] P. Bhatt, G. Shroff, and A. Misra. Dynamics of software maintenance. *ACM SIGSOFT Software Engineering Notes*, 29(4):1–5, 2004.

[4] D. Coleman, D. Ash, B. Lowther, and P. Oman. Using metrics to evaluate software system maintainability. *IEEE Computer*, 27(8):44–49, 1994.

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

[6] G. DiLucca, A. Fasolino, P. Tramontana, and C. Visaggio. Towards the definition of a maintainability model for web applications. In *Proceeding of the 8th European Conference on Software Maintenance and Reengineering*, pages 279–287. IEEE Computer Society Press, 2004.

[7] N. Fenton and S. Pfleeger. *Software Metrics A Rigourous and Practical Approach*. PWS, 2 edition, 1998.

[8] F. Fioravanti and P. Nesi. Estimation and prediction metrics for adaptive maintenance effort of object-oriented systems. *IEEE Transactions on Software Engineering*, 27(12):1062–1084, 2001.

[9] E. Ghosheh, J. Qaddour, M. Kuofie, and S. Black. A comparative analysis of maintainability approaches for web applications. In *Proceedings of the 4th ACS/IEEE International Conference on Computer Systems and Applications*, page 247. IEEE Computer Society Press, 2006.

[10] M. Kiewkanya, N. Jindasawat, and P. Muenchaisri. A methodology for constructing maintainability model of object-oriented design. In *Proceedings of the 4th International Conference on Quality Software*, pages 206–213. IEEE Computer Society Press, 2004.

[11] 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.

[12] T. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, 1976.

[13] E. Mendes, N. Mosley, and S. Counsell. Web metrics - estimating design and authoring effort. *IEEE Multimedia*, 08(01):50–57, 2001.

[14] E. Mendes, N. Mosley, and S. Counsell. Early web size measures and effort prediction for web costimation. In *Proceedings of the 9th International Software Metrics Symposium*, pages 18–39. IEEE Computer Society Press, 2003.

[15] D. Reifer. Web development: estimating quick-time-to-market. *IEEE Software*, 17(8):57–64, 2000.

[16] W. P. Stevens, G. J. Myers, and L. L. Constantine. Structured design. *IBM Systems Journal*, 13(2):115–139, 1974.