Volume 3, No. 5, May 2012
Journal of Global Research in Computer Science
ISSN-2229-371X
REVIEW ARTICLE
Available Online at www.jgrcs.info

SOME OBSERVATION ON MAINTAINABILITY METRICS AND MODELS FOR WEB BASED SOFTWARE SYSTEM

Anil Kumar Malviya[1], Laxmi Shanker Maurya*[2]
[1]Associate Professor,
Department of Computer Science and Engineering,
Kamla Nehru Institute of Technology, Sultanpur
anilkmalviya@yahoo.com
[2]Research Scholar,
Mewar University,
lsmaurya@yahoo.com

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 and to enhance maintainability, quantitative metrics for predicting web applications maintainability must be used. To estimate the maintenance cost and maintainability of software, many software metrics and models have been proposed in the literature. In most of these models researchers have focused on conventional software systems. Very few models are there for web based applications. In this paper we had tried to propose three primary level mathematical models of maintainability assessment for web based applications based on the studies conducted by Emad et al. [3], Heung et al. [2] and Silvia et al. [1].

Keywords: Web applications, metrics, maintainability, quantitative, model, observation.

INTRODUCTION

The World Wide Web has become a major delivery platform for a variety of complex and sophisticated enterprise applications in several domains. In addition to their inherent multifaceted functionality, these web applications exhibit complex behavior and place some unique demands on their usability, performance, security and ability to grow and evolve. However, a vast majority of these applications continue to be developed in an ad-hoc way, contributing to problems of usability, maintainability, quality and reliability.

While web development can benefit from established practices from other related disciplines, it has certain distinguishing characteristics that demand special considerations. In the recent years, there have been some developments towards addressing these problems and requirements. As an emerging discipline, **web engineering** actively promotes systematic, disciplined and quantifiable approaches towards successful development of high-quality, ubiquitously usable web-based systems and applications. In particular, web engineering focuses on the methodologies, techniques and tools that are the foundation of web application development and which support their design, development, evolution, and evaluation. Web application development has certain characteristics that make it different from traditional software, information system, or computer application development. Web engineering is multidisciplinary and encompasses contributions from diverse areas: systems analysis and design, software engineering, hypermedia/hypertext engineering, requirements engineering, human-computer interaction, user interface, information engineering, information indexing and retrieval, testing, modeling and simulation, project management, and graphic design and presentation. Web engineering is neither a clone, nor a subset of software engineering, although both involve programming and software development. While web Engineering uses software engineering principles, it encompasses new approaches, methodologies, tools, techniques, and guidelines to meet the unique **requirements of web-based applications.**

Maintainability

The maintainability is one of the critical aspects of a WA (Web Application): WAs have to be modified and evolve in a very fast way, then those features affecting it should be defined, identified and evaluated in order to improve/reduce the ones that have a positive/negative impact on the maintainability both during the development and maintenance process of a WA. Unfortunately, there are very few works in the literature addressing the problem of assessing the WA Maintainability.

Definition 1:-Maintainability is most commonly referred to as "the ease in which a system (for instance, a website or web application) can be modified or extended" (via Jeremy D. Miller).

Definition 2:- The ease with which repair may be made to the software as indicated by the following sub attributes: analyzability, changeability, stability and testability. (ISO 9126)

• Analyzability: how easy or difficult is it to diagnose the system for deficiencies or to identify the parts that need to be modified?

• Changeability: how easy or difficult is it to make adaptations to the system?

• Stability: how easy or difficult is it to keep the system in a consistent state during modification?

• Testability: how easy or difficult is it to test the system after modification?

Importance of maintainability

It has been measured that in the maintenance phase software professionals spend at least half of their time analyzing software to understand it [5]. The cost of software maintenance accounts for a large portion of the overall cost of a software system. Thus malfunctions of a critical software system can cause serious damages. For example, a problem in the Amazon.com web site in 1998 put the site down for several hours which cost the company an estimated $400,000. Also the relationship between the company and its customers can be greatly affected by such down time.
Quantitative metrics and models for predicting web applications' maintainability must be used to control the maintenance cost. The maintainability metrics and models can be useful in the following ways. First, predicting the maintenance and cost of maintenance tasks which helps in providing accurate estimates that can help in allocating the right project resources to maintenance tasks [9]. Second, comparing design documents which can help in choosing between different designs based on the maintainability of the design. Third, identifying the risky components of a software. Since some studies show that most faults occur on only few components of a software system. Fourth, establishing design and programming guidelines for software components. This can be done by establishing values that are acceptable or unacceptable and take actions on the components with unacceptable values. This means providing a threshold of software product metrics to provide early warnings of the system [10]. Fifth, making system level prediction where the maintainability of all components can be predicted by aggregating maintainability of single components. This can be used to predict the effort it will take to develop the whole software system [10].

The rest of the paper is organized as follows. Section 2 introduces web based systems. Section 3 demonstrates software metrics (both direct and indirect) along with importance of metrics in web application development. Section 4 gives an overview of related works in web application maintainability metrics and models. Section 5 describes certain observations on web application maintainability metrics and its relationship with maintainability and three proposed models by the authors for web application maintainability assessment. Section 6 focuses on results and discussion. Finally, section 7 provides our conclusions and future works.

WEB BASED SYSTEMS

A **web application** is an application that is accessed over a network such as the Internet or an intranet. The term may also mean a computer software application that is hosted in a browser-controlled environment (e.g. a Java applet) or coded in a browser-supported language (such as JavaScript, combined with a browser-rendered markup language like HTML) and reliant on a common web browser to render the application executable.

Web applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client, sometimes called a thin client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility. Common web applications include webmail, online retail sales, online auctions, wikis and many other functions.

SOFTWARE METRICS

IEEE Standard 1061 [8] lays out a methodology for developing metrics for software quality attributes. The standard defines an *attribute* as "a measurable physical or abstract property of an entity." A *quality factor* is a type of attribute, "a management-oriented attribute of software that contributes to its quality." A metric is a measurement function, and a software quality metric is "a function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality." To develop a set of metrics for a project, one creates a list of quality factors that are important for it.

Direct and indirect metrics

The IEEE Standard 1061 answer lies in the use of direct metrics. A *direct metric* is "a metric that does not depend upon a measure of any other attribute. Direct metrics are important under Standard 1061; because a direct metric is presumed valid and other metrics are validated in terms of it ("Use only validated metrics (i.e. either direct metrics or metrics validated with respect to direct metrics)"). "Direct" measurement is often used synonymously with fundamental" measurement [9] and contrasted with indirect or derived measurement [4]. The contrast between direct measurement and indirect, or derived measurement, is between a (direct) metric function whose domain is only one variable and a (derived) function whose domain is an n-tuple. For example, density is a function of mass and volume. Some common derived metrics in software engineering are:
• Programmer productivity (code size/ programming time)
• Module defect density (bugs / module size)
• Requirements stability (number of initial requirements /total number of requirements)
• System spoilage (effort spent fixing faults / total project effort)\

Four examples of direct measurement provided by Fenton & Pfleeger:
• *Length* of source code (measured by lines of code);
• *Duration* of testing process (measured by elapsed time in hours);
• *Number of defects discovered* during the testing process (measured by counting defects);

• *Time* a programmer spends on a project (measured by months
worked).

Importance of metrics in web application development

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. Metrics or measures play fundamental role in overall assessment of the quality of the software systems in general and web based systems in particular. Henceforth, specialized metrics should be investigated to assess the quality characteristics of the web applications particularly the maintainability.

RELATED WORKS

Almost no studies have been made towards establishing a sound definition and validation of metrics for early measuring the structural complexity of web applications. For instance, Dhyani *et al*. present a survey that classifies a wide set of web metrics [5]. Some of them were proposed to measure web graph properties1, addressing the structural complexity of web applications. However, the majority of these metrics have not been accepted in practice because they were not built using a clearly defined process for defining software measures. There are a myriad of design recommendations and guidelines for building usable Web sites. These guidelines address a broad range of Web site features, from the amount of content on a page to the breadth and depth of pages in the site. However, there is little consistency and overlap among them making it difficult to know which guidelines to adhere to. Furthermore, there is a wide gap between a heuristic and its operationalization in metrics.

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. Guiseppe et al. [6] measured maintainability by measuring both modifiability and understandability. Coleman quantifies maintainability via its Maintainability Index. The Maintainability Index is measured as a function of directly measurable attributes *A*1 through *A*n as shown in Equation 1:

$$M = f(A1, A2... An) \quad (1)$$

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) [9] 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 [10] 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 [11] 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.

OBSERVATIONS

Observation1

There are several design quality attributes defined in the literature that have an effect on maintainability such as coupling, cohesion and complexity. Emad et al. [3] has defined metrics for the following design attributes: size, complexity, coupling and reusability. The metrics are shown below.

Table1: Web Application Design Metrics

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)

This study aims to answer the following question: Is there a relationship between the metrics identified in Table1 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 (M) maintainability sub characteristic. 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. In the result Emad et al. [3] has found that all the four hypotheses were accepted. Now, we can interpret these findings in the form of following mathematical equations. Such as:

$$M \propto \frac{1}{Size} \quad \text{--- (1)}$$

$$M \propto \frac{1}{Structural\ Complexity} \quad \text{--- (2)}$$

$$M \propto \frac{1}{Coupling} \quad \text{--- (3)}$$

$$M \propto \text{Reusability} \quad \text{--- (4)}$$

Combining above 4 equations we get the following equation

$$M \propto \frac{Reusability}{Size \times Structural\ Complexity \times Coupling} \quad \text{--- (5)}$$

$$\text{Or, } M = k \frac{Reusability}{Size \times Structural\ Complexity \times Coupling} \quad \text{--- (6)}$$

Equation 6 may be assumed as a maintainability assessment model based on the used metrics.
The value of constant k may be calculated on the basis of various environmental and human factors.

Observation 2

For estimating maintenance effort, Heung et al. [2] choose four object-oriented metrics: RFC [3], LCOM [3], DAC [9], and LOC. Software metric can be largely classified into the following three categories: cohesion, coupling, and size. In the experiment, one representative metric for each category is selected. More specifically, C&K's LCOM [3] is used for cohesion measuring metrics, and C&K's RFC [3] and Li and Henry's DAC [9] are used for coupling measuring metrics. Finally, LOC is used for size measuring metrics. In addition, LCOM, RFC, DAC and LOC have been used in empirical studies for estimating various quality attributes such as maintainability, fault-proneness and productivity. The used metrics can be summarized as follows:

- **LCOM(Lack of Cohesion in Methods)**, one of C& K metrics suite, is a representative metric for measuring cohesiveness of a class and have been widely used for several experiments. LCOM is defined as follows: LCOM = |P|-|Q|, if |P| > |Q|. Otherwise, LCOM = 0. |Q| denotes the number of method pairs which refer to instance variables commonly used in a class. |P| denotes the number of method pairs that have no shared instance variables. As can be seen in the definition, LCOM is a reverse metric for cohesion, in other words, the higher LCOM is, the worse the cohesion is.

- **RFC(Response for Classes)** denotes the total number of methods which are responding to some class objects or some classes invoke to. Accordingly, RFC is a coupling metric representing dependency relationship

between classes. In general, the higher a coupling is, the more the maintenance activity costs.

- **DAC(Data Abstraction Coupling)** means dependencies between classes on the basis of data abstraction. More specifically, it denotes the number of classes related to a class with aggregations.

- **LOC(Line of Code)** is used to estimate a system size based on the number of statement lines in a program. It regards all statements except blank and comments as part of program size. Although LOC gives a big difference in developing similar software according to programming languages, or programmer's coding styles, it is traditionally used because of its simplicity and ease of intuitive understanding.

The observed relationship between Maintenance Effort (M_e) and used object-oriented metrics by Heung et al. [2] may be interpreted in the form of mathematical equations as follows:-

$M_e \propto RFC$ --- (1)

$M_e \propto DAC$ --- (2)

$M_e \propto LCOM$ --- (3)

$M_e \propto LOC$ --- (4)

We also know that there is inverse relation between M_e and Maintainability (M). So, this relationship can be interpreted as:-

$M_e \propto \frac{1}{M}$ ---(5)

and hence, the relationship between Maintainability(M) and the object-oriented metrics presented in above equations may be interpreted as follows:-

$M \propto \frac{1}{RFC}$ ---(6)

$M \propto \frac{1}{DAC}$ ---(7)

$M \propto \frac{1}{LCOM}$ ---(8)

$M \propto \frac{1}{LOC}$ ---(9)

Combining equations 6, 7, 8 and 9 we get the following equation

$M \propto \frac{1}{RFC \times DAC \times LCOM \times LOC}$ ---(10)

Or, $M = k\frac{1}{RFC \times DAC \times LCOM \times LOC}$ ---(11)

Now, Equation 11 may be used as another maintainability model based on the used metrics and the value of the constant k may be estimated based on various environmental and personnel factors.

Observation 3

While modeling the navigational structure of a web application, several aspects should be taken into account such as the underlying structure of navigation (i.e., how the navigation space is organized); which objects will be navigated, as well as what are the effects of a navigation action. According to the OOWS (Object-Oriented Web Solutions) approach, the navigation space is organized by defining a unique navigational map for each agent [7]. Silvia et al. [1] used following metrics for web application maintainability assessment. Table 2 shows some metrics for navigational maps based on the morphological characteristics of the navigational model. NNC and NNL metrics can be used as indicators of the navigational model size. Also, these metrics can be used as an indicator of density of a navigational map (DNM). This can be calculated as:

$D_{NM} =$ NNL/NNC (1)

The *Depth of a Navigational Map* (DNM) is just the distance of a root navigational context to a leaf context. It indicates the ease with which the target navigational context can be reached and the likely importance of its content. The interpretation is: the larger the distance of a leaf navigational context from the root, the harder it is for the agent to reach this context and potentially the less important this context will be in the map. We can measure the maximum, minimum and average depth of a navigational map for each agent. The *Breadth of a Navigational Map* (BNM) is the number of exploration navigational contexts (i.e. at the first level of contexts). The interpretation is: the larger the number of exploration navigational contexts, the harder it is for the agent to understand the web application (too many options at once). Even web applications with a non-deep hierarchical structure and a reduced number of navigational contexts may have a complex navigation structure.

Table 2.Some Metrics for Navigational Maps

Metric Name	Metric Definition
Number of Navigational Contexts (NNC)	The total number of navigational contexts in a navigational map.
Number of Navigational Links (NNL)	The total number of navigational Links (NNL) in a navigational map.
Density of a Navigational Map (DeNM)	An indicator of density of a navigational map.
Depth of a Navigational	The longest distance of a root navigational context

Map (DNM)	to a leaf context.
Breadth of a Navigational Map (BNM)	The total number of exploration navigational contexts.
Minimum Path Between Navigational Contexts (MPBNC)	The minimum amount of navigational links that are necessary to transverse from a source to a target navigational context.
Number of Path Between Navigational Contexts (NPBNC)	The amount of alternative paths in order to reach two contexts within a navigational map.
Compactness (Cp)	The degree of interconnectivity of a navigational map.

The *Compactness* metric (Cp) represents the edge-to-node ratio attribute. It refers to the degree of interconnection among nodes pertaining to a hypermedia graph. Applying this metric to navigational maps, the degree of interconnectivity of navigational contexts is obtained. For example, when there are few links in a navigational map, navigational contexts could be difficult to reach by following links. In addition, users may become disoriented because they need to go through many steps to get some piece of information. Some parts of a navigational map may not even be connected by links at all (known as orphan or dead-end nodes).
The interpretation for this metric is: high compactness means that each navigational context can easily reach any other node in the map (this may indicate a highly connected map which can lead to disorientation). Low compactness may indicate an insufficient number of links, which may mean that parts of a map are disconnected. Table 3 shows some metrics for Navigational Contexts. Traditionally, one commonly used metric is the fan-in / fan-out metric [8], which is based on the idea of coupling. The interpretation is that the larger the number of couples, the greater the degree of interdependence and difficulty of maintenance, and the lower the potential for reuse.

Table 3. Some Metrics for Navigational Context.

Metric Name	**Metric Definition**
Fan-In of a Navigational Context (FINC)	This counts the number of invocations a navigational context calls.
Fan-Out of a Navigational Context (FONC)	This counts the number of navigational links that call a navigational context.
Number of Navigational Classes (NNCl)	The total number of classes within a navigational context.
Number of Attributes (NA)	The total number of attributes of all classes in a navigational context.
Number of Methods (NM)	The total number of methods of all classes in a navigational context.

His presented metrics represent quantitative measures of navigational properties. For instance these metrics permit a structural analysis, and reveal potential navigational problems such as unnecessary circular (link) paths and dead-end nodesT.
The relationships identified by Silvia et al. [1] in their study between size and structural complexity and maintenance time(T_m) and maintainability(M) may be denoted mathematically in the form of following equations:-
$T_m \propto$ size ---(1)

and
$T_m \propto$ structural complexity ---(2)

Combining equations 1 and 2 we get
$T_m \propto$ size×structural complexity ---(3)

Or, T_m = k(size)×(structural complexity) ---(4)
The value of the constant k may be evaluated on the basis of various environmental and personal factors.

Further we know that there is inverse relationship between T_m and M and hence the above equations (1), (2), (3) and (4) may be represented as follows:-
$M \propto \frac{1}{size}$ ---(5)

And

$$M \propto \frac{1}{structural\ complexity} \quad \text{---(6)}$$

Combining equations 5 and 6 we get

$$M \propto \frac{1}{size \times structural\ complexity} \quad \text{---(7)}$$

$$\text{Or, } M = k\frac{1}{size \times structural\ complexity} \quad \text{---(8)}$$

The metrics used in the study may lead another model for maintainability assessment denoted by equation 8. The value of the constant k may be calculated based on various environmental and personal factors involved in the system.

RESULTS AND DISCUSSION

In section 5 from observation 1, 2 and 3 respectively we get the first, second and third maintainability assessment models represented by equations (1), (2) and (3) as follows:-

$$M = k\frac{Reusability}{Size \times Structural\ Complexity \times Coupling} \quad \text{--- (1)}$$

$$M = k\frac{1}{RFC \times DAC \times LCOM \times LOC} \quad \text{--- (2)}$$

$$M = k\frac{1}{size \times structural\ complexity} \quad \text{---(3)}$$

In first maintainability assessment model four design attributes: size, structural complexity, coupling and reusability has been used. In the second model four object oriented metrics RFC, LCOM, DAC, and LOC has been used. RFC is a coupling metric representing dependency relationship between classes. LCOM is a reverse metric for cohesion, in other words, the higher LCOM is, the worse the cohesion is. DAC (Data Abstraction Coupling) means dependencies between classes on the basis of data abstraction. LOC is the lines of code. In the third model size and structural complexity has been used as the metric.
The value of constant k in all the three models may be estimated based on various personnel and environmental factors involved in the process of web application development.

CONCLUSION & FUTURE WORK

The cost of software maintenance accounts for a large portion of the overall cost of a software system. Therefore, we need to effectively manage software maintenance activities. As in the conventional software systems, we can apply measurement based approach to estimating and predicting maintenance efforts. 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. In Observation1 metrics proposed by Emad et al. [3] for the following design attributes: coupling, complexity, size, and reusability were used to introduce a primary model for maintainability assessment which is depicted in equation 6. In Observation2 four object-oriented metrics have been proposed by Heung et al. [2] such as LCOM, RFC, DAC and LOC for assessment of maintenance effort. Using these metrics we have proposed another model for the maintainability assessment which is depicted by equation 11. In Observation3 eight metrics for Navigational Maps and five metrics for Navigational Context has been proposed by Silvia et al. [1]. Using these metrics we have identified the third model for maintainability assessment which is depicted by equation 8. One drawback of our work is that there is no concrete clue for the evaluation of the constant k . Our future work will try to focus on determining the relevancy and extent of the factors which are affecting k such that accurate estimation of it may be ensured. Future work may also include comparative study of design and code level maintainability metrics for web based applications.

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 SoftwareMetrics Symposium, pages 200–210. IEEE Computer Society Press, 2003.

[2] H. Seok Chae, T. Yeon Kim, Woo-Sung Jung, Joon-Sang Lee. Using Metrics for Estimating Maintainability of Web Applications: An Empirical Study. In Proceedings of the 6th International Conference on Computer and Information Science, pages 1053 - 1059. IEEE Computer Society Press, 2007.

[3] Emad Ghosheh, S. Black and J. Qaddour. Design metrics for Web application maintainability measurement. IEEE/ACS International Conference on Computer Systems and Applications, Pages 778 – 784. IEEE Computer Society Press, 2008.

[4] Sanjeev Dhawan and R. Kumar. Analyzing performance of Web based metrics for Evaluating reliability and maintainability of hypermedia applications. 3rd International Conference on Broadband Communications, Information Technology & Biomedical Applications, Pages 376 - 383. IEEE Computer Society Press, 2008.

[5] Cem Kaner, Walter P. Bond. Software Engineering Metrics: What Do They Measure and How Do We Know? 10th International software metrics symposium, Pages 1-12 . IEEE Computer Society Press, 2004.

[6] Guiseppe Antonio Di Lucca, A. R. Fasolino, P. Tramontana and C. A. Visaggio. Towards the definition of a maintainability model for Web applications. In the proceedings of the Eighth European Conference on Software Maintenance and Reengineering (CSMR'04), Pages 279-287, IEEE Computer Society Press, 2004.

[7] E. Ghosheh, S. Black, and J. Qaddour. An industrial study using UML design metrics for web applications. In Computer and Information Science, volume 131 of Studies in Computational Intelligence, chapter 20, pages 231–241. Springer-Verlag, 2008.

[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] 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.

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

[11] 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.

SHORT BIODATA OF ALL THE AUTHORS

Anil Kumar Malviya is an Associate Professor in the Department of Computer Science and Engineering, at Kamla Nehru Institute of Technology, Sultanpur. He is PhD in Computer Science from B. R. Ambedkar University, Agra.

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 Pantnagar, M. Tech. in Information Technology from AAIDU Allahabad and MBA in HR from IGNOU New Delhi.