Towards Benchmarking of Traditional Software Quality Metrics on Web-based Systems

Olaye, Edoghogo, Apeh, Simon T.
Department of Computer Engineering
University of Benin
Benin City, Nigeria
E-mail: edoghogho.olaye@uniben.edu; apeh@uniben.edu
Phone: +2348061227175

ABSTRACT

Software quality is a very important topic in software engineering and the use of software metrics is an accepted method of estimating software quality. Several metrics have been proposed over the years leaving software developers with many choices but no scientific way of determining the right metrics for a specific software. The current approach is largely based on intuition, popularity, expert opinions and availability of tools. We propose a traditional software quality benchmark system for web-based system called TSQM Benchmark. The concept of the research and the benchmark architecture are presented in this paper as a foundation for this research. This paper also describes the design methodology and preliminary development. When the TSQM Benchmark is fully implemented, it will establish a method to determine the degree of suitability of traditional software quality metrics for evaluating web-based systems. The benchmark should be able to correctly rank traditional software metrics in the order of which metric is most suitable for the assessment of web-based system quality of a given category. This study will contribute to researchers and software engineering practitioners in the area of web engineering with a systematic, unbiased and scientific evaluation of software metrics.

Keywords: Benchmarking, Software Quality Metrics, Web-based systems, Web applications.

1. BACKGROUND TO THE STUDY

Software quality metrics are means of measuring specific attributes of a software artefact such as source codes. The attributes measured could serve as an indicator of the quality of the system, productivity of the developers or efficacy of the design method. Today, business have become practically online and relies on software such as web-based systems to make products and services available to a global audience via the Internet (Desai & Srivastava, 2012). A web-based system or web application is a dynamic extension of the World Wide Web or an application server (Nourie, 2006). Despite the many benefits of web-based systems, its use has created problems for users and developers. Quality remains a challenging problem amongst other problems such as privacy, security, user acceptance, and cost.

A subset of the quality problem is how to define quality and how to measure quality of a web based system. Software engineers in general and web engineers in particular are evolving methodologies, tools, standards and frameworks to solve the quality issue. Web engineering is a holistic approach that deals with all aspects of Web-based systems development, starting from conception and development to implementation, performance evaluation, and continual maintenance (Ginige & Murugesan, 2001). Its main focus is the establishment and use of sound scientific and engineering management principles as well as systematic approaches for high-quality web-based systems (Heuser, 2004). The advocacy for web engineering according to Mendes (2005) are geared towards a common terminology, more empirical studies, development of professional web engineers, research networks and special interest groups. (Mendes, 2005).

Empirical studies involves experimentation on software systems to collect data on real-life projects for the formulation and validation of hypothesis concerning software engineering methods (Sommerville, 2011). Software metrics and measurement are the basis of empirical software engineering (Endres & Rombach, 2003). Therefore, there is the need to develop software metrics to support the evidence based approach of empirical software engineering. In this regards, metrics were introduced in the 1970s for quantitative software quality measurements (Boehm, Brown, & Lipow, 1976). However, no success was made to develop as single metric for quality (Mills & Shingler, 1988). Since then, several metrics have been proposed as software development methodologies evolved. These generation of metrics can be roughly categorized into traditional software metrics, (Mills, 1988), object-oriented software metrics (Shaik & Reddy, 2012) and web metrics (Calero, Ruiz, & Piattini, 2005), (Dhyani, Ng, & Bhowmick, 2002).

The number of metrics in each category are increasing drastically in the order presented and many new metrics have been introduced deserving their own category. For example, a study reported 133 Web Size metrics for web sites alone within 12 years (Mendes, Counsell, & Mosley, 2005) while another survey identified more than 27 dynamic metrics (Chhabra & Gupta, 2010). Studies have shown that traditional software metrics are inadequate for object-oriented software

(Basili, Briand, & Melo, 1996). This claim directly led to the development of OO metrics such as the CK Metrics suite (Chidamber & Kemerer, 1994).

However, many developers still use traditional software metrics for object-oriented software products (Tegarden, Sheetz, & Monarchi, 1992) and web based systems. A typical example is the lines of code (LOC) metric that is being collected by majority of software metrics tools (Lokhande, 2012).

In the area of web-systems or web-application development the use of scripting programming languages such as PHP and JavaScript is prevalent. Though many of these scripting languages are being reengineered to make them object-oriented, it appears that many web-based application developers still use the procedural and functional programming languages. Accordingly, though the state of the art is towards web metrics, many web-based application developers still use myriad of metrics that cuts across traditional, object-oriented and web metrics. At the same time, there are numerous metrics being developed for web-based systems with no clear cut was of choosing the best set of metrics for a given software product. Efforts made in this direction have not addressed the problem. Some of these efforts include: development of a catalogue of software metrics (Bouwers, Deursen, & Visser, 2014), metrics evaluation (Schackmann, Jansen, Lischkowitz, & Lichter, 2009) and a pluggable tool for metrics evaluation (Higo*, et al.*, 2011).

The current reality is such that there is no established way of judging which metric is better than the other for quality assessment of web-based systems. Thus the primary focus of the current research geared towards developing a standard basis for selecting and applying traditional software metrics which is the most popular and most common on web-based systems. The benchmarking method was selected as the judge that will judge the suitability of the existing traditional judges (traditional metrics).

The authours wish to emphasize that benchmark results are not presented in this paper but will be published elsewhere. This paper only reports on the architecture of the benchmarking method. The architecture has been implemented to a stage that its practicality can be assessed.

2. STATEMENT OF THE PROBLEM

The manner in which web based software systems are developed, deployed, and managed affects quality. Web software developers often use haphazard approaches which lack rigor, systematic techniques, sound methodologies, and quality assurance (Ginige & Murugesan, 2001). Traditional software metrics are sufficient for traditional software development such as desktop software and operating systems. These metrics are robust and have been applied to many software systems with good results. However, the relatively more recent web based metrics though numerous are arguable laboratory based. They are mainly proposed for use on new web based systems rather than for existing web based systems.

It may appear that the best practice for quality evaluation of existing web-based systems is to apply traditional metrics on web based software or to use aspects of the metrics.

However, there is no established bases to determine which of the metrics are best suited for this purpose. In this regards, very few benchmarks studies have been conducted and currently, metrics selection is mostly arbitrary. The specific problems this research seeks to address are as follows:

i. No established bases to determine which traditional metrics are best suited for web-based system.
ii. Very few software metrics benchmarks studies have been conducted and currently, metrics selection is mostly arbitrary

3. MOTIVATION AND KEY OBJECTIVES

The goal of this research is to benchmark traditional software quality metrics on web based systems to bridge this gap with the following objectives:

i. To develop a benchmarking framework for traditional software metrics on web-based software systems.
ii. To identify the weakness or sufficiency of applying existing traditional software quality metrics on web-based systems.
iii. To develop a benchmarking tool for traditional software metrics on web based systems
iv. To generate benchmark data for traditional software quality metrics on web-based systems
v. To validate the benchmark data generated

The study encompasses benchmarking activities using software tools. It involves development of new tools and modification of existing tools. The benchmarking is limited to traditional software quality metrics that can be statically determined from source codes. The high-level objective is achieved through the following research questions.

RQ1: What is the basis for choosing a software metrics over another when developing web based systems?

RQ2: How can a metric be compared against another metrics to determine which of the two is a better measure of a specific quality characteristic for web based software system?

3.1 Relevance of the Study

The current research work is necessitated upon the following developments: The growth of internet usage, proliferation of web-based systems, the need for quality software, the effect of poor quality on web-based systems, problem with web metrics and the need for a benchmarking study. These developments and state-of-the-art creates numerous challenges for software developers and software users. A software practitioner described the problem as poor quality levels due to "*shortage of solid empirical data about quality, productivity, schedules, costs, and how these results vary based on development methods, tools, and programming languages*" (Jones, 2014b). Software failure leads to losses due to the increasing reliance on software, especially web based software over the years. Software failure is a function of software quality. The internet

is being used for a variety of reasons ranging from health (Pereira, Koski, Hanson, Bruera, & Mackey, 2000), marketing (Brown & Goolsbee, 2000) and mobile organization management (Sołtysik-Piorunkiewicz, 2015).
As of 2014 the software industry labors under a variety of non-standard and highly inaccurate measures compounded by very poor measurement practices (Jones, 2014a). Studies have shown that internet traffic have doubled each year and there is an insatiable want for more bandwidth (Odlyzko, 2003). Developers often use ad hoc, hacker-type approaches, which lack rigor, systematic techniques, sound methodologies, and quality assurance (Ginige & Murugesan, 2001).

Similarly, some metrics not well defined and are not empirically or theoretically validated, and hence they can confuse interested users instead of helping them (Calero, et al., 2005). In addition, there is no single metric to measure quality, few synthetic metrics and inadequate metric tools. Benchmarking searches for improvement and best practices but is often used as a software evaluation method for comparing software systems (García-Castro, 2008).

3.2 Related Work
3.3 Related work on Benchmarking
Benchmarking is a process of running a number of standard tests using alternative tools/methods (usually tools) and assessing the relative performance of the tools against those tests.

Benchmarking is a term that has different meanings in different disciplines. It generally refers to a comparison of an organization's performance or product's performance against its peers. Clearly, the term was used long before the invention of computers (Jones, 2010). Several definitions have been proposed for benchmarking:

Kitchenham, Linkman (1997) viewed benchmarking as a method of evaluating software tools. In this context benchmarking was defined as *"..a process of running a number of standard tests using alternative tools/ methods (usually tools) and assessing the relative performance of the tools against those tests" (Kitchenham, Linkman, & Law, 1997)*

Sim *et al*, (2003) defined benchmarking as a test or set of tests used to compare the performance of alternative tools or techniques. They advocated for the definition of benchmarks in software engineering areas (Sim, Easterbrook, & Holt, 2003).

Gracia-Castro argues that benchmarking offers more benefits than evaluation through continuous improvement and recommendations on best practices (García-Castro, 2008). Blackburn *et al*., (2006) argues that benchmarks drive computer science research and industry product development (Blackburn, *et al*., 2006). It sets standards for innovation and can encourage or stifle it. (Runapongsa, Patel, Jagadish, & Al-Khalifa, 2002) worked on benchmarking database systems. They indicated that benchmarks are valuable to potential users of a database system in providing an indication of the performance that the user can expect on their specific application. (Sim, *et al*., 2003) noted that benchmarks have been used in computer science to compare the performance of computer systems, information retrieval algorithms, databases, and many other technologies.
However, they argued that some benchmarking is not straightforward in software engineering because the performance measures are not straightforward but can be quite complex. They suggested that the complexity arises from the intention of the tools and techniques for the creation of large software systems. A tool called the YCSB Client, to execute the YCSB benchmarks was developed by Cooper et al, (2010). The purpose of YCSB is to tackle the lack of applesto- apples performance comparisons that makes it difficult to understand the tradeoffs between Cloud systems and the workloads for which they are suited (Cooper, Silberstein, Tam, Ramakrishnan, & Sears, 2010). Peacekeeper is a free and fast browser test that measures the speed of a web browser. It works on any computer, device or platform that is equipped with a web browser ("Peacekeeper,").

A modified version of Peacemeaker is PCMark 8, that implements industry standard PC benchmarking tools. The TPC-C benchmark simulates a complete computing environment where a population of users executes transactions against a database. The benchmark is focussed on the principal activities (transactions) of an order-entry environment. The SPEC Web 2009 benchmark is the next-generation SPEC benchmark for evaluating web server performance. Workloads include: Banking, which is a fully secure SSL-based workload; Ecommerce, which includes both SSL and non-SSL requests; and Support, which is a non-SSL workload that includes large downloads (SPEC, 2009). The DaCapo Benchmarks, a Java Benchmarking Development and Analysis research is a set of open source, client-side Java benchmarks focused on improving methodologies for choosing and evaluating benchmarks to foster innovation in system design and implementation for Java and other managed languages. It covers 6 metrics namely: WMC, DIT, NOC, CBO, RFC, LCOM. DaCapo improves over SPEC Java in a variety of ways, including more complex code, richer object behaviours, and more demanding memory system requirements (Blackburn, et al., 2006).

Rentrop, (2006) investigated the uses of software metrics as benchmarks for source code (Rentrop, 2006). Dolan & Moré, 2002 attempted to benchmark optimization software with performance profiles based on sound theory and composite graph. Dolan & Moré, (2002) generated the benchmark results by running a solver on a set P of problems and recording information of interest such as the number of function evaluations and the computing time. Maxwell & Forselius, (2000) studied benchmarking software development productivity towards comparing a company's software-development productivity to that of similar projects. They developed benchmarking equations derived from a productivity-variation analysis performed on the Experience database (Maxwell & Forselius, 2000).

Other benchmarking research that are not specifically meant for software metrics include Benchmarking Attribute Selection Techniques for Discrete Class Data Mining (Hall & Holmes, 2003), and Benchmarking Semantic Web technology which sought to develop a methodology and apply it to

benchmark interoperability of semantic web technologies using RDF and OWL as interchange languages (Castro, 2008), (García-Castro & Gómez-Pérez, 2010).
Similarly, Vorhies and Morgan, (2005) used benchmarking in a marketing research by empirically examining the potential business performance benefits available from benchmarking the marketing capabilities of top-performing firms (Vorhies & Morgan, 2005). None of the studies reviewed attempted to benchmarks software quality metrics. The only studies that come close are that of Rentrop and Blackburn et al, 2006. In the current research, a method will be developed to directly map the existing benchmarks for popular framework into the developed benchmark. Similarly, a generic benchmark for frameworks will be developed with the purpose of integrating framework that have not been benchmarked.

3.4 Web Software Quality and Web Evaluation

Software quality concerns are not new. As early as the 1960s, quality issues were being investigated with initial thrust to define quality (Boehm*, et al.*, 1976). Quality is viewed from the perspective of the general utility of software. The description of quality is somewhat hierarchical. Quality attributes such as consistency is a sub attribute of reliability which in turn is a sub attribute of as-is-utility. In the characteristic tree, there are 15 attribute specified in the Boehm model.

Several researches have been conducted in the area of web metrics. Dhyani *et al*. (2002) have proposed a web classification framework to try to determine how the classified metrics can be applied to improve web information access and use. The work of Dhyani *et al.* (2002) however did not consider important dimensions such as life-cycle processes and web features. To provide , a broader classification, the Web Quality Model (WQM) was developed which distinguishes three dimensions related to web features, life-cycle processes and quality characteristics (Ruiz *et al*., 2003).
The first version of the WQM model was developed in 2003 and was refined in a survey (Calero et al., 2004) by using it in the classification of 326 web metrics. Further work on the model attempted to refine it to support the classification of metric related to effort and reuse. This was achieved by including organizational life-cycle processes (Ruhe, Jeffery, & Wieczorek, 2003).

The usefulness of quality models was investigated in (Al-Kilidar, Cox, & Kitchenham, 2005) and a study of how such models can be used in software package selection investigated in (Franch & Carvallo, 2003). Usability evaluation methods can be mainly classified into two groups: empirical methods and inspection methods. Empirical methods are based on observing, capturing, and analyzing usage data from real end users, while inspection methods are performed by expert evaluators or designers, and are based on reviewing the usability aspects of Web artifacts such as conceptual models or user interfaces with regard to their conformance with a set of guidelines. (Abrahão, Insfran, & Fernandez, 2014)

Gupta, Goyal, & Goyal proposed a Hierarchical Model for Object-oriented Design Quality Assessment (HMOOD-QA) model for determining the quality of a product built by using object-oriented approach. They considered a 4-layered (basic, metric, factor and quality) product quality based on the factors viz. Defect Density, Complexity and Change Effort (Gupta, Goyal, & Goyal, 2015).
The test specific quality model is a quality model for test specification is an adaptation of ISO/IEC 9126 to the domain of test specification. While the ISO/IEC 9126 model deals with internal quality, external quality, and quality in use, the model focuses on internal quality characteristics (Zeiss, Vega, Schieferdecker, Neukirchen, & Grabowski, 2007). Other Models (Kavindra, Praveen, & Jitendra, 2014) include: a user-centric web quality assessment model presented by Nakwichian and Sunetnanta. Similarly, Brajnik suggested the adoption of Goal-Question-Metric paradigm as a useful framework to guide the definition of the quality model (Brajnik, 2001). Fitzpatrick et al argued for models based on Human Computer Interaction standards (Fitzpatrick, 1999).

Olsina et al described a Quality Evaluation Model and outlined a quality requirement tree which provides a descriptive framework to specify these quality characteristics (Luis Olsina & Rossi, 2000) (Luiz Olsina, 1999). After evaluating some models, Kavindra, et al submitted that while they are suitable for internal and external evaluation, the models did not covers all quality aspects especially communication aspects such as theoretical and specific aspects and even more important, aesthetic aspects. They argued that since the quality model of a website is determined by the process of evaluation, design, implementation and validation involving a variety of methods and tools. In order to carry out on these processes, quality metrics need to be defined (Kavindra*, et al.*, 2014).

3.5 Related work on Metrics

A metric is defined as a measure of degree to which a software system possesses or exhibits a certain (quality) characteristics. According to Chauhan, Gupta, & Dixit (2014) , there are three types of metrics; Process metrics, project metrics and product metrics. There are many other ways to categorize software metrics. One way is to consider the phase at which the metrics are collected in the software development lifecycle. Planning metrics can be collected before development while code metrics can only be collected during or after coding. Another way to classify metrics is to consider if the metric is static or dynamic (Mayo, Wake, & Henry, 1990). Some categories traditional metrics considered by some authors include: size metrics (Lines of code, Function points, bang), complexity metrics (cyclometric complexity, extensions to v(G), knots), Halstead's product metrics (program vocabulary, program length, program volume) and quality metrics (defect metrics, reliability metrics, maintainability metrics) (Mills & Shingler, 1988). A metric suit for object oriented design was developed by (Chidamber & Kemerer, 1994).

Work has been carried out to investigate the suitability of benchmarking based on software metrics method to determine the maintainability of the source code of software systems (Rentrop, 2006). The state of the art in metrics is towards more empirical validations (Srinivasan & Devi, 2014) and integration of metrics tools (Jain, Srivastava, & Katiyar, 2014). Metrics tools have been developed to present metrics in UML class diagrams (Turan, 2015). Using metrics from service level agreements (SLAs) to monitor quality of web services (Andreasen, Nielsen, Schrøder, & Stage, 2015). The trend towards making software usable have led to the introduction of more usability metrics (Abrahão, *et al.*, 2014). The usability beam has also landed on open source software (Andreasen, et al., 2015). Metrics selection for improving the performance of software has also been given consideration of recent (V. Chauhan, *et al.*, 2014). Metrics for software quality estimation during early stages of software developing is receiving focus by some researchers (R. Chauhan, Singh, Saraswat, Joya, & Gunjan, 2014)

3.6 Related work on Metric Tools

Many tools such as OOMeter (Jain, *et al.*, 2014), CKJM (Spinellis, 2005), and a Metric Based Code Analyzing Tool (Fernando, *et al.*, 2012) have been developed for collecting software metrics. A review of some of these tools can be found in the work of (Lincke, Lundberg, & Löwe, 2008) and (Novak & Rakić, 2010). Two tools (QMetric and MASU) deserves special mention because they form part of the proposed TSQM Benchmark. The QMetric tool suite (Figure 1) provides a generic evaluation engine for evaluating process metrics, as well as tool support for the definition of quality assessment models and their automatic evaluation (Schackmann, *et al.*, 2009).

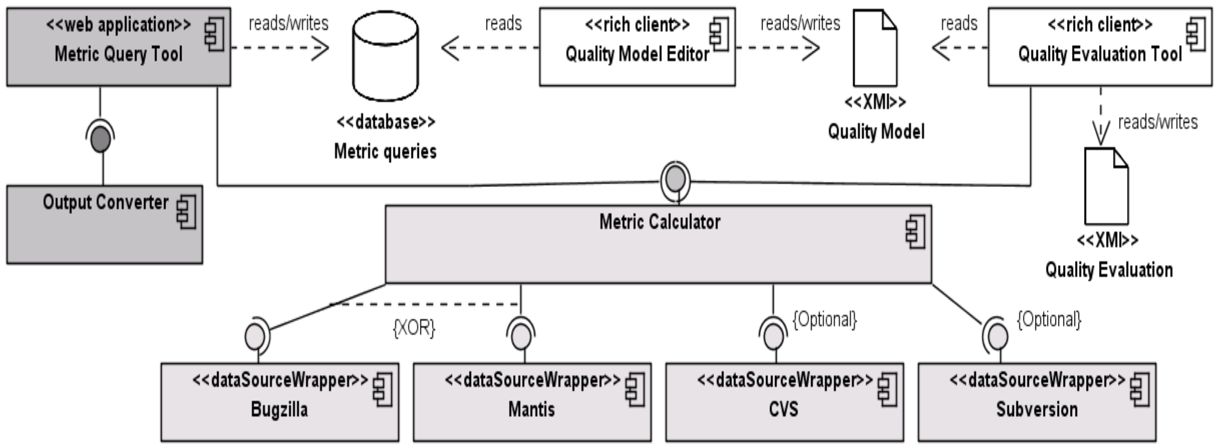

Figure 1: QMetric tool suit Architecture

The Metrics Assessment plugin platform for Software Unit (MASU) is a measurement tool developed by Japanese scholars using the Java programming language (Higo, *et al.*, 2011). MASU can handles Java full grammar and the code is under development. The architecture of MASU is shown in Figure 2.

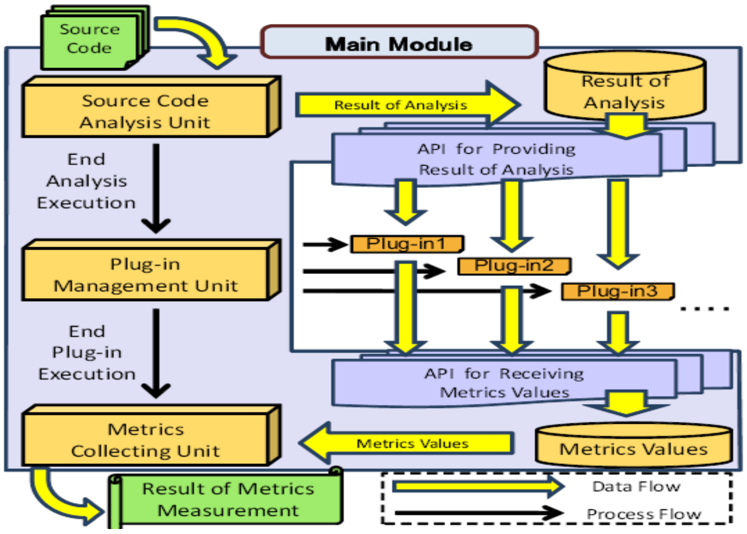

Figure 2: Architecture of MASU

These two tools collectively offers two major distinctive features: (1) The ability to collect any metric using a single tool by extending its functionality; and (2) the ability to evaluate the quality of a software product using a standard quality model.

4. Research Methodology

4.1 The Research Design

The research objectives was accomplished through the process described in Figure 3.

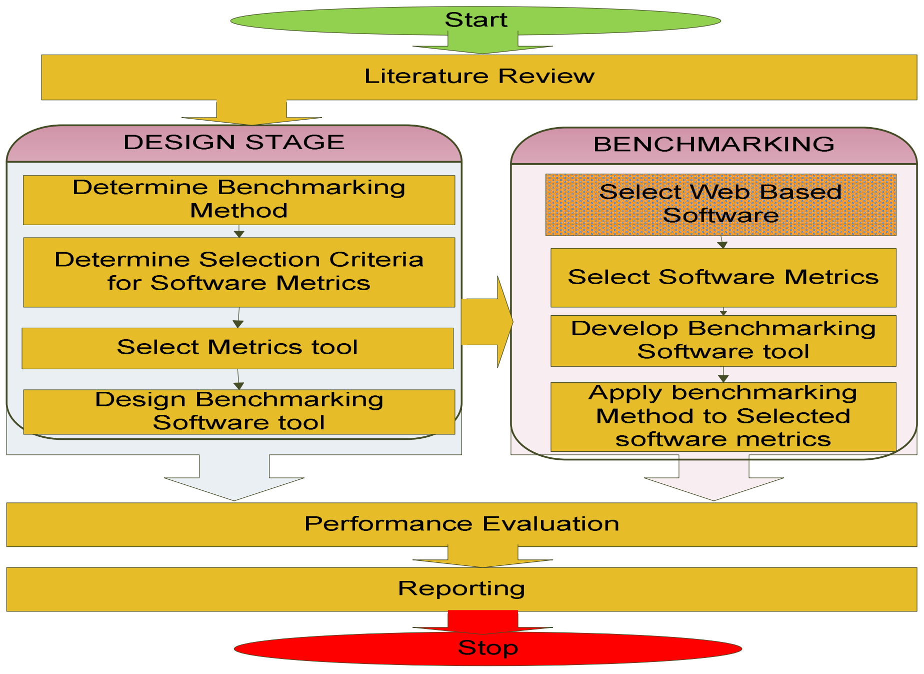
Start
Literature Review
DESIGN STAGE
Determine Benchmarking Method
Determine Selection Criteria for Software Metrics
Select Metrics tool
Design Benchmarking Software tool
BENCHMARKING
Select Web Based Software
Select Software Metrics
Develop Benchmarking Software tool
Apply benchmarking Method to Selected software metrics
Performance Evaluation
Reporting
Stop

Figure 3: TSQM research process

4.2 Design Workflow for the TSQM Benchmark

The design workflow is broken down into two phases. Each phase consist of a set of activities that will be performed iteratively until the desired outcome is achieved. Phase 1 has been concluded and Phase 2 is ongoing.

Phase 1 (Specification and design of the TSQM Benchmark): This phase of design specifies the materials and the proposed techniques that will be used to achieve the current research objectives. The activities in this stage includes: Determination of evaluation criteria for traditional software quality metrics and Determination of selection criteria for web-based systems, Application of selection criteria to select 10 web based systems; Application of selection criteria to select traditional software quality metrics, Determination of selection criteria of metrics tools, Development of the architecture for the TSQM Benchmark and Setting up the benchmarking Laboratory.

Phase 2 (Development and implementation of the TSQM Benchmark): This phase is concerned with the development of the Traditional Software Quality Metrics (TSQM) method and actual benchmarking. Activities include: Development of the TSQM Benchmark tool, Quality evaluation of selected web-based system to establish a base line and generate benchmark queries; Generate benchmark queries, Evaluate selected software quality metrics, Evaluate the selected benchmarking methods, Benchmark traditional software metrics using existing tools and evaluation methods, Apply benchmarking method to selected software quality metrics.

4.2.1 Metrics Selection Criteria

The selection criteria for traditional software quality metric is that the metrics must be a metric that was originally developed to collect measure quality attributes of software written using a procedural programming language. The metrics should also be a classic metric in that it must have been widely studied by different authors.

4.2.2 Web-based System Selection Criteria

Software selection would be based on a number of factors, the most important factor being that they should be web based. Other factors that will be considered include: availability of design artifacts, availability of source codes, licenses, programming language, popularity, industry type, diversity of sizes of the systems, user base, framework used, platform, web technologies used and , diversity of functionality provided. Preference is given to web-based systems whose source code is readily available without license limitations. The software source codes were obtained from GitHub (www.github.com), SourceForge, vendor websites and Google codes.

It was observed that most source codes were being migrated to GitHub. The benchmarking method proposed in the current research will be greatly facilitated by source code version information available from GitHub. The preliminary software selection include:

- Adminer, Single source RDMS for MySQL
- phpMyAdmin, A database management system for MySQL
- RoundCube, A full featured email client
- SquirrelMail, A full featured email client http://squirrelmail.org/download.php
- Disqus,
- OSQA, Question and Answer system
- Converse.JS, An XMPP chat client that can be directly integrated into a website
- wolfCMS, Content management system
- Wordpess, Requires installation
- Joomla, Requires installation
- moodle, e-learning application. Requires PHP version 5.4.4. Requires installation
- WaeUP, An e-education portal
- Clumsy Bird, A MelonJS made "Flappy Bird" clone
- Justice (Also a tool), Creates an on page toolbar that displays page timing metrics and a streaming fps meter.
- Phantomas (Also a tool), PhantomJS-based modular web performance metrics collector

4.3 Design Considerations and component selection for Benchmarking Tool

The design goal of the benchmarking process is to provide a level playing field for all the selected traditional software quality metrics. To achieve this goal, a framework for an extensible interface layer for each metric in the benchmarking tool is developed. Implementation of this layer will be in form of plug-in that gets the value of a metric and formats the data to a type required by the benchmarking tool. The metric value will be validated with results obtained by existing metric tools for the metric under investigation.

The benchmarking tools that will be used in the current research will be developed from the following components which are part of other tools:

i. MASU : Metrics Assessment plugin platform for Software Unit. Although it is developed for object oriented programming language, some of its components and plugin structure will be used for the benchmarking tool. The source code analysis unit will be modified and adapted as well as the plugin management unit
ii. QMetrics: It is a tool designed for generic evaluation of process metrics. The tool uses XML to define quality evaluation as well as metrics definitions. These approaches will be utilized in developing the benchmarking tool in the current research.
iii. ANTL: ANother Tool for Language Recognition is a parser generator for reading, processing, executing, or translating structured text or binary files.

4.4 Proposed benchmarking method for the TSQM-Benchmark

The method used by TSQM-Benchmark is similar to the metric validation method described in (Gafni, 2008) which seeks to prove that metrics behave in a consistent and logical mode in quantifying the quality of software. In the current research, traditional software metrics values will be calculated for the selected web based software systems. Then the web software system is deliberately and scientifically altered to reduce its quality along each of the quality dimensions. The metrics is then recalculated and new value noted. Ranking for each metric for the "TSQM Benchmark" is calculated based on the degree of change and the consistency of change. A rank factor is determined which is a function of the old and new metric values as expressed in Equation 3.1. The flowchart for the TSQM Benchmark method is shown in Figure 4.

$$rankFactor = f(OldMetricVal - NewMetricVal)$$

The ranking will be based on the following statistical analysis of experimental results:

A. Which metrics behave in the most consistent and logical mode?
B. To what extent does a metric M behave in a more consistent and logical mode than metric Mi in quantifying web-based system quality?
C. To what extent does the value for metric Mi on a given web-based system change when the quality of the web based software system increased or decreases.

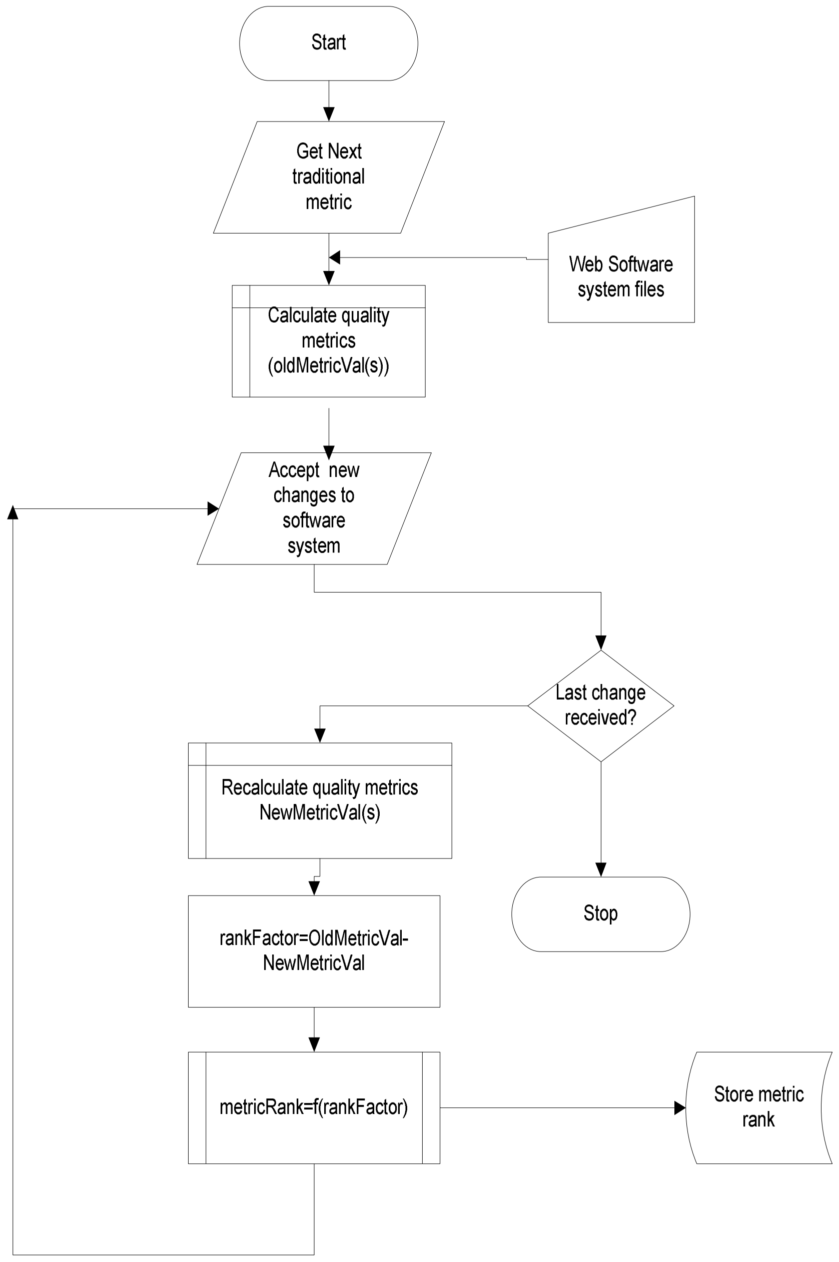
Start
Get Next traditional metric
Web Software system files
Calculate quality metrics (oldMetricVal(s))
Accept new changes to software system
Last change received?
Recalculate quality metrics NewMetricVal(s)
Stop
rankFactor=OldMetricVal-NewMetricVal
metricRank=f(rankFactor)
Store metric rank

Figure 4: Flow chart for proposed TSQM Benchmark

4.5 Setting up the Benchmarking Laboratory

The benchmarking laboratory will be setup in such a way that it can support both the development and the implementation of the benchmarking tool. The lab will contain systems with configurations to support the running of the selected benchmarking tools. The major components are the main computer, the support computer, Ethernet switch and KVM switch. The Main Test Computer has an Intel Core i7 processor (2.3GHz), 8GB RAM, Windows 7 Home Basic (64 bits) operating system, Java runtime environment (jre) , and Java development kit (jdk). the support computer (Test Computer 2) has an Intel Duo Core processor (2.3GHz), 6GB RAM, Windows 8 (64 bits) operating system, Java runtime environment (jre) , Java development kit (jdk).

5.1 The TSQM Benchmark

The architecture of the benchmarking system is shown in Figure 5. It is named Traditional Software Quality (TSQM) Benchmark. The proposed benchmarking method will have the capacity to evaluate traditional software quality metrics based on their suitability for web-based systems quality evaluation.

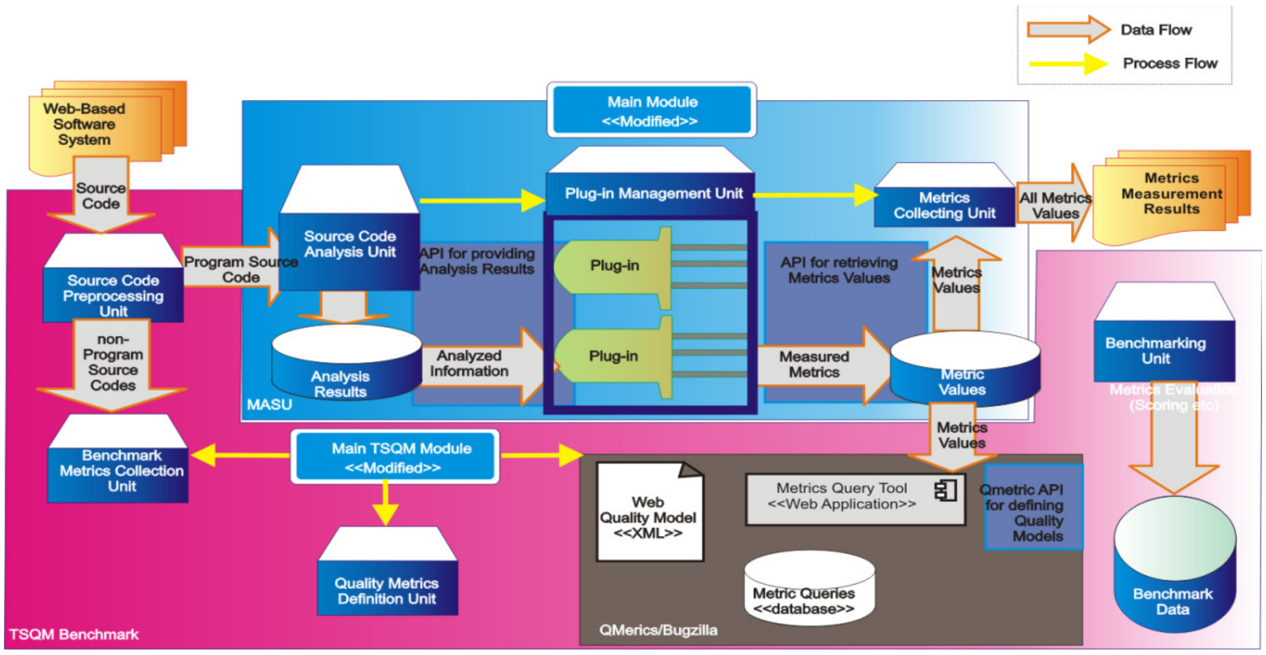
Data Flow
Process Flow
Web-Based Software System
Source Code
Main Module <<Modified>>
Source Code Analysis Unit
Plug-in Management Unit
Metrics Collecting Unit
All Metrics Values
Metrics Measurement Results
Program Source Code
API for providing Analysis Results
Plug-in
API for retrieving Metrics Values
Metrics Values
Source Code Preprocessing Unit
non-Program Source Codes
Analysis Results
Analyzed Information
Plug-in
Measured Metrics
Metric Values
Benchmarking Unit
Metrics Evaluation (Scoring etc)
MASU
Metrics Values
Benchmark Metrics Collection Unit
Main TSQM Module <<Modified>>
Web Quality Model <<XML>>
Metrics Query Tool <<Web Application>>
Qmetric API for defining Quality Models
Quality Metrics Definition Unit
Metric Queries <<database>>
Benchmark Data
TSQM Benchmark
QMerics/Bugzilla

Figure 5: The proposed Benchmarking Architecture

The system comprises a code analysis unit, metrics plug-in unit and quality model integration unit. The code analysis is done to separate web data and HTML related information from the actual web-application program. The second stage of code analysis is the conventional parsing to make processing easy. The selected traditional metrics will be implemented as plugins which will be managed by the plug-in management unit. All the metrics are collected for a given web-based system and stored in the metrics value database. The WebQ quality model(Luis Olsina & Rossi, 2002) will be encoded in XML and used to develop metric queries by the Metric query tool. These queries are stored in the metric queries database. The benchmarking unit will implement the benchmarking method to rank the metrics based on encoded quality model information and metric values for all the selected web-based systems. The results of the benchmarking will be stored in the benchmark database.

5.2 Data presentation and Expected Results

Preliminary results have been obtained from source code analysis of the selected web-based systems. The analysis of the result for presentation is currently ongoing. It is expected that at the end of the current research, the TSQM-Benchmark method will be fully developed, applied on selected traditional software metrics using selected web-based systems and evaluated. A number of specific benchmark results are expected for every given metrics such as Functionality Benchmark, Reliability Benchmark, Usability Benchmark, Efficiency Benchmark, Portability Benchmark, and Maintainability Benchmark. The benchmark results will be presented in a table similar to Table 1. Using the table, all the traditional software quality metrics under investigation will be ranked in terms of how well the given metrics can estimate the quality of a web-based system.

Table 1: Format for Benchmark Result

Metrics	**Functionality (Sore)**	**Reliability (Score)**	**Usability (Score)**	**Efficiency (Score)**	**...**
Metric 1	0.0	0.0	0.0	0.0	...
Metric 2	0.0	0.0	0.0	0.0	...
...					
...					
Metrics N	...	...	...	...	...

6. DISCUSSION AND FINDINGS

The expected outcome of this research is that it will establish a method to determine the degree of suitability of traditional software quality metrics in the evaluation of web-based systems. This study will contribute to researchers and software engineering practitioners in the area of web engineering with a systematic, unbiased and scientific evaluation of software metrics. Software metrics selection will be made easier.
The preliminary findinds that may require further investigations are as follows:

i. Most web-based systems were not designed to meet any specific quality standard
ii. Very few web-based system developers are aware that there are numerous web metrics
iii. There are few unified metric tools for collecting software metrics for web-based systems

7. CONCLUDING REMARKS

This paper has described the research concept, methodology and expected result of a proposed research that entails the benchmarking of traditional software quality metrics on web-based systems. The research seeks to solve the problem of how best to evaluate the quality of web-based systems using existing traditional software quality metrics instead of the current haphazard metric development and selection practice.
Through the analysis of the research problem, the proposed methodology is expected to yield results that will demonstrate that the objectives are reached. The proposed method, when implemented will provide insight into which traditional software metrics is best suited for web-based systems. The implication of the expected result (ranked metrics) is that software developers can easily choose which benchmark to use and which to ignore. Future metrics can also be subjected to this benchmarking method to rightly determine the relative ranking to the metrics that will be ranked in this research.

7.1 Future work

Our future work will involve the full implementation of the TSQM benchmark in line with the objectives presented in this paper. There are several different directions that future work in this area can continue. Firstly, further work is required in the area of benchmarking web metrics. This work could extend the current research. Secondly, work needs to be done to develop a system that will make it easy for web-based system developers to adhere to standards. They need to be aware that these standards exists and if possible have a simple tool to check the adherence level of software projects. Finally, this study did not cover metrics for Agile methods considering the trend of creative thinking (Crawford, Barra, Soto, Misra, & Monfroy, 2013). Quality evaluation remains important, therefore new methods need to be evolved to accommodate the design methodology.

REFERENCES

1. Abrahão, S., Insfran, E., & Fernandez, A. (2014). Designing Highly Usable Web Applications.
2. Al-Kilidar, H., Cox, K., & Kitchenham, B. (2005). *The use and usefulness of the ISO/IEC 9126 quality standard.* Paper presented at the Empirical Software Engineering, 2005. 2005 International Symposium on.
3. Andreasen, M. S., Nielsen, H. V., Schrøder, S. O., & Stage, J. (2015). Usability in open source software development: Opinions and practice. *Information technology and control, 35*(3).
4. Basili, V. R., Briand, L. C., & Melo, W. L. (1996). A validation of object-oriented design metrics as quality indicators. *Software Engineering, IEEE Transactions on, 22*(10), 751-761.
5. Blackburn, S. M., Garner, R., Hoffmann, C., Khang, A. M., McKinley, K. S., Bentzur, R., *et al.* (2006). *The DaCapo benchmarks: Java benchmarking development and analysis.* Paper presented at the ACM Sigplan Notices.
6. Boehm, B. W., Brown, J. R., & Lipow, M. (1976). *Quantitative evaluation of software quality.* Paper presented at the Proceedings of the 2nd international conference on Software engineering.
7. Bouwers, E., Deursen, A. v., & Visser, J. (2014). *Towards a catalog format for software metrics.* Paper presented at the Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics.
8. Brajnik, G. (2001). *Towards valid quality models for websites.* Paper presented at the Proceedings of the 7th Conference on Human Factors and the Web.
9. Brown, J. R., & Goolsbee, A. (2000). *Does the internet make markets more competitive?* : National Bureau of Economic Research.
10. Calero, C., Ruiz, J., & Piattini, M. (2005). Classifying web metrics using the web quality model. *Online Information Review, 29*(3), 227-248.
11. Castro, R. u. G. ı. (2008). *Benchmarking Semantic Web technology.* Universidad Polit´ecnica de Madrid.
12. Chauhan, R., Singh, R., Saraswat, A., Joya, A. H., & Gunjan, V. K. (2014). Estimation of Software Quality using Object Oriented Design Metrics. *International Journal of Innovative Research in Computer and Communication Engineering, 2*(1), 2581-2586.
13. Chauhan, V., Gupta, D. L., & Dixit, S. (2014). Role of Software Metrics to Improve Software Quality. *Complexity, 2*(3), 6.
14. Chhabra, J. K., & Gupta, V. (2010). A survey of dynamic software metrics. *Journal of computer science and technology, 25*(5), 1016-1029.
15. Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on, 20*(6), 476-493.
16. Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., & Sears, R. (2010). *Benchmarking cloud serving systems with YCSB.* Paper presented at the Proceedings of the 1st ACM symposium on Cloud computing.
17. Crawford, B., Barra, C. L. d. l., Soto, R., Misra, S., & Monfroy, E. (2013). Creative Thinking in eXtreme Programming.
18. Dhyani, D., Ng, W. K., & Bhowmick, S. S. (2002). A survey of Web metrics. *ACM Computing Surveys (CSUR), 34*(4), 469-503.
19. Endres, A., & Rombach, D. (2003). *Empirical Software Engineering: A Handbook of Observations, Laws and Theories*. Harlow, UK: Addison-Wesley.
20. Fernando, T., Fernando, M., Senevirathne, A., Amarasinghe, N., Indraraj, D., & Kodagoda, N. (2012). Metric Based Code Analyzing ToolT.
21. Fitzpatrick, R. (1999). *A Process for Appraising Commercial Usability Evaluation Methods, Human-Computer Interaction: Ergonomics and User Interfaces.* Paper presented at the HCI International.
22. Franch, X., & Carvallo, J. P. (2003). Using quality models in software package selection. *Software, IEEE, 20*(1), 34-41.
23. Gafni, R. (2008). Framework for quality metrics in mobile-wireless information systems. *Interdisciplinary Journal of Information, Knowledge, and Management, 3*, 23-38.
24. García-Castro, R. (2008). *Benchmarking Semantic Web Technology.* Informatica.
25. García-Castro, R., & Gómez-Pérez, A. (2010). Interoperability results for Semantic Web technologies using OWL as the interchange language. *Web semantics: science, services and Agents on the World Wide Web, 8*(4), 278-291.
26. Ginige, A., & Murugesan, S. (2001). Web engineering: An introduction. *MultiMedia, IEEE, 8*(1), 14-18.
27. Gupta, N., Goyal, D., & Goyal, M. (2015). A hierarchical model for object-oriented design quality assessment. *International Journal of Advances in Engineering Sciences, 5*(3), 1-6.
28. Hall, M. A., & Holmes, G. (2003). Benchmarking attribute selection techniques for discrete class data mining. *Knowledge and Data Engineering, IEEE Transactions on, 15*(6), 1437-1447.
29. Heuser, L. (2004). The real world or Web engineering? *Web Engineering* (pp. 1-5): Springer.
30. Higo, Y., Saitoh, A., Yamada, G., Miyake, T., Kusumoto, S., & Inoue, K. (2011). *A pluggable tool for measuring software metrics from source code.* Paper presented at the Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA).
31. Jain, S., Srivastava, V., & Katiyar, P. (2014). Integration of Metric Tools for Software Testing. *International Journal of Enhanced Research in Science Technology & Engineering, ISSN*, 2319-7463.
32. Jones, C. (2010). *Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies*: McGraw-Hill Companies.
33. Jones, C. (2014a). The Mess of Software Metrics Retrieved from http://namcookanalytics.com/mess-software-metrics/

34. Jones, C. (2014b). Sources of software benchmarks. Retrieved from http://namcookanalytics.com/wp-content/uploads/2013/07/SOURCES-OF-SOFTWARE-BENCHMARKS-24.pdf
35. Kavindra, K. S., Praveen, K., & Jitendra, M. (2014). Implementation of a Model for Websites Quality Evaluation-DU Website. *International Journal of Innovations & Advancement in Computer Science IJIACS, 3*(1).
36. Kitchenham, B., Linkman, S., & Law, D. (1997). DESMET: a methodology for evaluating software engineering methods and tools. *Computing & Control Engineering Journal, 8*(3), 120-126.
37. Lincke, R., Lundberg, J., & Löwe, W. (2008). *Comparing software metrics tools.* Paper presented at the Proceedings of the 2008 international symposium on Software testing and analysis.
38. Lokhande, R. N. (2012). Software Engineering Metrics: Introduction.
39. Maxwell, K. D., & Forselius, P. (2000). Benchmarking software development productivity. *Software, IEEE, 17*(1), 80-88.
40. Mayo, K. A., Wake, S. A., & Henry, S. M. (1990). Static and Dynamic Software Quality Metric Tools.
41. Mendes, E. (2005). *A systematic review of Web engineering research.* Paper presented at the Empirical Software Engineering, 2005. 2005 International Symposium on.
42. Mendes, E., Counsell, S., & Mosley, N. (2005). Towards a taxonomy of hypermedia and web application size metrics *Web Engineering* (pp. 110-123): Springer.
43. Mills, E. E. (1988). *Software metrics*: DTIC Document.
44. Mills, E. E., & Shingler, K. H. (1988). Software Metrics-SEI Curriculum Module SEI-CM-12-1.1.
45. Nourie, D. (2006). Java Technologies for Web Applications. *Java.* Retrieved from http://www.oracle.com/technetwork/articles/java/webapps-1-138794.html
46. Novak, J., & Rakić, G. (2010). *Comparison of software metrics tools for: net.* Paper presented at the Proc. of 13th International Multiconference Information Society-IS, Vol A.
47. Odlyzko, A. M. (2003). *Internet traffic growth: Sources and implications.* Paper presented at the ITCom 2003.
48. Olsina, L. (1999). *Web-site quality evaluation method: a case study on museums.* Paper presented at the Proceedings of the ICSE.
49. Olsina, L., & Rossi, G. (2000). Web Engineering: A Quantitative Methodology for Quality Evaluation and Comparison of Web Applications. *The Electonic Journal of the Argentine Society for Informatics and Operations Research, 3*(1).
50. Olsina, L., & Rossi, G. (2002). Measuring Web application quality with WebQEM. *Ieee Multimedia*(4), 20-29.
51. Peacekeeper. from http://peacekeeper.futuremark.com/run.action
52. Pereira, J. L., Koski, S., Hanson, J., Bruera, E. D., & Mackey, J. R. (2000). Internet usage among women with breast cancer: an exploratory study. *Clinical breast cancer, 1*(2), 148-153.
53. Rentrop, J. (2006). *Software Metrics as Benchmarks for Source Code Quality of Software Systems.* Universiteit van Amsterdam.
54. Ruhe, M., Jeffery, R., & Wieczorek, I. (2003). *Using web objects for estimating software development effort for web applications.* Paper presented at the Software Metrics Symposium, 2003. Proceedings. Ninth International.
55. Runapongsa, K., Patel, J. M., Jagadish, H., & Al-Khalifa, S. (2002). *The Michigan Benchmark: A microbenchmark for XML query processing systems.* Paper presented at the EEXTT.
56. Schackmann, H., Jansen, M., Lischkowitz, C., & Lichter, H. (2009). *QMetric-a metric tool suite for the evaluation of software process data.* Paper presented at the ICSE Companion.
57. Shaik, A., & Reddy, K. (2012). Object oriented software metrics and quality assessment: Current state of the art. *International Journal of Computer Applications, 37*(11).
58. Sim, S. E., Easterbrook, S., & Holt, R. C. (2003). *Using benchmarking to advance research: A challenge to software engineering.* Paper presented at the Proceedings of the 25th International Conference on Software Engineering.
59. Sołtysik-Piorunkiewicz, A. (2015). Web technology in mobile organization management: Innowacje w zarządzaniu i inżynierii produkcji, Opole: PTZP.
60. Sommerville, I. (2011). *Software Engineering* (9th ed.): Addison-Wesley.
61. SPEC. (2009). SPEC Web 2009. from Standard Performance Evaluation Corporation: https://www.spec.org/web2009/
62. Spinellis, D. (2005). Tool writing: A forgotten art? *IEEE Software, 22*(4), 9–11.
63. Srinivasan, K., & Devi, T. (2014). Software Metrics Validation Methodologies in Software Engineering. *International Journal of Software Engineering & Applications, 5*(6), 87.
64. Tegarden, D. P., Sheetz, S. D., & Monarchi, D. E. (1992). *Effectiveness of traditional software metrics for object-oriented systems.* Paper presented at the System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on.
65. Turan, M. (2015). Integrating Software Metrics with UML Class Diagrams. *Lecture Notes on Software Engineering, 3*(3), 220.
66. Vorhies, D. W., & Morgan, N. A. (2005). Benchmarking marketing capabilities for sustainable competitive advantage. *Journal of marketing, 69*(1), 80-94.
67. Zeiss, B., Vega, D., Schieferdecker, I., Neukirchen, H., & Grabowski, J. (2007). Applying the iso 9126 quality model to test specifications. *Software Engineering, 15*(6), 231-242.