Selection, Classification and Computation of an Optimal Set of Software Metrics for Requirement, Design and Testing Phases of SDLC

Shefali Singla1 , Dr. Dheerendra Singh2, Yogesh Verma3, Dr. R. NandaKumar4
1, 2Shaheed Udham Singh College of Engg, Tangori,Mohali, Punjab, India
3,4 Space Applications Centre, Ahmedabad, India
shefalis51@gmail.com, professordsingh@gmail.com, yogeshverma@sac.isro.gov.in, nandakumar@sac.isro.gov.in

ABSTRACT
This paper presents a comparative study is on testing and design tools such as Sonar Qube, Simple code and Eclipse metric. This delivers tool-dependent metrics results and has even implications on the results of analyses based on these metrics results. In short, the metrics based assessment of a software system and measures taken to improve the quality of software. To support our case, we conducted an experiment with a 11 free metrics tools. We found that we get different metrics form open source tools. By analyzing the results of all free source tools we conclude that Sonar Qube provides best result from all open source tools.

Keywords: Software metrics, requirements metrics, design metrics, testing metrics, open source tools, sonar.

1. INTRODUCTION
Accurate measurement is a prerequisite for all engineering disciplines, and software engineering is not an exception. For decades seek engineers and researchers to express features of software with numbers in order to facilitate software quality assessment. A large body of software quality metrics have been developed, and numerous tools exist to collect metrics from program representations. This large variety of tools allows a user to select the tool best suited, e.g., depending on its handling, tool support, or price. The definition of the metrics we first selected refers to the standard literature. During implementation, we found that the metrics definitions are ambiguous and there are lots of metrics which is used to improve the quality of software, as we are working on the three phases of software development life cycle SDLC, we make classification of metrics according to their category, here we provide its description, formula, category refer in the literature study part. After classification of metrics, we have selected open source tools which are used to compute this metrics.

2. METRICS SELECTION
We composed a list of metrics based on the literature survey from various sources like web and papers to select a set of metrics for requirements, design and testing phase of software development. These metrics were further analysed and categorized from an organization point and converted in to an optimal set of metrics.

Under requirement phase , total 27 quantitative metrics were selected and categorized under sever different categories.

Table-1 Summary of Requirement Metrics

<table>
<thead>
<tr>
<th>SN</th>
<th>Category Name</th>
<th>No of selected metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Productivity</td>
<td>4</td>
</tr>
<tr>
<td>2.</td>
<td>Quality</td>
<td>6</td>
</tr>
<tr>
<td>3.</td>
<td>Volatile</td>
<td>3</td>
</tr>
<tr>
<td>4.</td>
<td>Understand ability</td>
<td>4</td>
</tr>
<tr>
<td>5.</td>
<td>Traceability</td>
<td>2</td>
</tr>
<tr>
<td>6.</td>
<td>Effort</td>
<td>3</td>
</tr>
<tr>
<td>7.</td>
<td>Miscellaneous</td>
<td>5</td>
</tr>
</tbody>
</table>

In table 1 , there are 4 metrics for the productivity of an software, 6 metrics which gives the value of quality, 3 metrics which tells the volatility of an software, 4 metrics estimates whether the given software is understandable or not, and the 3 metrics give insight about how much effort will be consumed while work on that particular software.

Some of the important metrics are requirements traceability, changed, Unambiguous, consistency and not redundant.

Similarly in design phase, in total 42 metrics were considered as per the table 2. Here, 14 metrics specify that how many no of components depend upon the input of another component, 4 metrics defines about the reusability of metrics, 1 metric is about the progress of the software and 4 metrics are the package metrics.

The set of metrics includes no of methods, McCabe’s complexity, No of attributes, no of children, abstractness, cohesion, coupling etc.


Table-2 Summary of Design Metrics

<table>
<thead>
<tr>
<th>SN</th>
<th>Category Selected metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Component Interaction Metrics 14</td>
</tr>
<tr>
<td>2.</td>
<td>Connector Metrics 2</td>
</tr>
<tr>
<td>3.</td>
<td>Composite Metrics 1</td>
</tr>
<tr>
<td>4.</td>
<td>Reusability Metrics 4</td>
</tr>
<tr>
<td>5.</td>
<td>Functional Coverage Metrics 4</td>
</tr>
<tr>
<td>6.</td>
<td>Progress Metric 1</td>
</tr>
<tr>
<td>7.</td>
<td>Package Metric 4</td>
</tr>
<tr>
<td>8.</td>
<td>Coupling metric 2</td>
</tr>
<tr>
<td>9.</td>
<td>Un Categorized 10</td>
</tr>
</tbody>
</table>

In testing phase, total 46 metrics were collected, analysed and categorized under different categories as per table -3

Table-3 Summary of Testing Metrics

<table>
<thead>
<tr>
<th>SN</th>
<th>Category</th>
<th>No of metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Defect</td>
<td>15</td>
</tr>
<tr>
<td>2.</td>
<td>Effort</td>
<td>4</td>
</tr>
<tr>
<td>3.</td>
<td>Efficiency</td>
<td>3</td>
</tr>
<tr>
<td>4.</td>
<td>Quality</td>
<td>9</td>
</tr>
<tr>
<td>5.</td>
<td>Traceability</td>
<td>1</td>
</tr>
<tr>
<td>6.</td>
<td>Stability</td>
<td>1</td>
</tr>
<tr>
<td>7.</td>
<td>Progress</td>
<td>12</td>
</tr>
<tr>
<td>8.</td>
<td>Productivity</td>
<td>1</td>
</tr>
</tbody>
</table>

The set of metrics consist of defect density, defect pattern, time to fix a defect, no of defects, test effort percentage, test efficiency, quality of fixes and so on.

3. OPTIMAL SET OF METRICS

The table 4 presents the optimal set of metrics of three phases based on the literature survey

Table-4 List of optimal metrics for Organization

<table>
<thead>
<tr>
<th>Phase</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements phase</td>
<td>Metric Name, Uniqueness, Correctness, Changed Requirements, Misinterpreted Requirement, Understandable Requirements, Modifiable, Traced, Requirement Testing SRS Quality, Unambiguous, Complete, Understandable, Verifiable, Internal Consistent, Annotated By Relative Importance, Annotated By Relative Stability, Annotated By Version, Precise, Traceable, Not redundant, At Right level of detail Organized, Achievable, Electronically stored, concise, Design independent, reusable</td>
</tr>
<tr>
<td>Design phase</td>
<td>Total Number of Component (TNC), Average No of Methods Per Component (ANMC), Total Number of Implemented Components (TNIC) Total Number of Links (TNL), Average no of Links Between Component (ANLC), Width of the Composition Tree(WCT), Component dependency Metrics, Component Interaction dependency Metrics (CIDM), Size of the Architecture, Component</td>
</tr>
<tr>
<td>Testing phase</td>
<td>Density of Architecture, Average Distance of Architecture, Component Customization Metrics, Component Reusability Metrics, Existence of Meta Information(EMI), Rate of Component Observability (RCO), Rate of Component Customizability (RCC), Self Completeness of Component’s Parameter, Function Coverage, Compliance Familiarity, Support, Proximity Metric, Data Model Compliance Index (DCMI), Functional Model Compliance Index (FMCI), Component Usage Strength (CUS), Component Backward Dependency Strength (CBDS), Abstractness, Instability, Distance from the Main Sequence, Attribute Hiding Factor (AHF), Method Hiding factor (MHI), Polymorphism Factor (PF) Structure Complexity of a Module, Data Complexity of Module, Module Coupling Indicator Mc, Revised Coupling Metric, Function Point, Bang Metric</td>
</tr>
</tbody>
</table>

4. OPEN SOURCE TOOLS EVALUATION & METRIC COLLECTION

Further to automate metrics collection task, following set of tools were considered from open source domain particularly for design and implementation phase:

I. Sonar
II. CCCC
III. Chidamber &Kemerer Java Metrics(ckjm)
IV. Eclipse Metrics Plug-in 1.3.6
V. Simple Code Metrics
VI. Squale
VII. SourceMonitor
VIII. LocMetrics
IX. LineTally
X. Unified Code Count
XI. Universal Code Line Counter
Further, Tools were installed, configured and evaluated up to the criteria set by the organization i.e. set of metrics.

Tools gives us the metrics namely Lines of Code (LOC), Number of methods (NOM), McCabe’s Complexity (MV), Weighted Methods per class (WMC), Lack Of Cohesion in Methods (LCOM), Afferent Coupling (Ca), Efferent Coupling (Ce), Number of static methods (NSM), Normalized Distance (RMD), Specialized Index (SIX), Instability (RMI), Number Of Overridden methods (NORM), Number Of Interfaces (NOI), Number Of Parameters (PAR), Abstractness (RMA), Nested Block depth (NBD), Number of static attributes (NSF), Number of Classes (NOC), Method Lines of Code (MLOC), Documented API, Undocumented API, Number of Assessors, Number Of Files (NOF), Complexity per function, Lines of Comments (COM), Cyclomatic complexity per class, Cyclomatic complexity per file, Line of Imports, Number of Public Methods (NPM), Number of Source files (NSou), Number of Statements, Percentage of Duplicate Lines, Duplicated Blocks. A total set of around 38 metrics involving various phases of software development life cycle.

Out of all the evaluated tools, Sonar gives us a good set of metrics as well rich features from testers, developers, and management point of view.

5. CONCLUSION & FUTURE WORK

Based upon the literature study, the list of software metrics for requirement, design and testing phases has been evaluated, selected and classified to be used as listed:

A set of 27 initial metrics are selected for the first phase (Requirement Phase), among them there are 4 productivity metrics, 6 quality metrics, 3 volatile metrics, 4 understandability metrics, 2 traceability metrics, 3 effort metrics and 5 miscellaneous metrics.

A set of 42 unique metrics are selected for the second phase (Design Phase), among them there are 14 component interaction metrics, 2 connector metrics, 1 composite metrics, 4 reusability metrics, 4 functional coverage metrics, 1 proximity metric, 4 package metrics, 2 coupling metrics, 10 uncategorized. A set of 46 unique metrics are selected for the third phase (Testing Phase), among them there are 15 for defect metrics, 4 effort metrics, 3 efficiency metrics, 9 quality metrics, 1 traceability metrics, 1 stability metrics, 11 progress metrics and 1 productivity metrics.

In this research a comparative study is done on testing tools such as Sonar Qube, simple code (Net beans plug-in) and Eclipse metrics plug-in 1.3.6, for computing the optimal set of metrics for requirement, design and testing phase of software development life cycle. Sonar Qube computes the 30 optimal metrics, where the Eclipse metrics plug-in 1.3.6 computes 23 optimal metrics.
simple code (Net beans plug-in) computes 5 optimal metrics. Sonar has a provision for integration with other plugins / tools namely configuration management, PDF report generation and JMeter Testing. Sonar Qube gives the graphical representation which tells about the critical, major and minor issues of metrics whereas, rest of two not show the graphical representation of metrics that are only text based. Further, we find more results on open source tools for computing, evaluating and presenting of metrics at each phase of software development life cycle (SDL).

6. ACKNOWLEDGMENTS
The author is highly grateful to Dr. Manjit Singh Grewal, Director Shaheed Udham Singh College of Engineering and Technology (SUSCET), Tangori (Mohali), for providing this opportunity to carry out the present thesis work. I would like to express a deep sense of gratitude and thanks profusely to my Supervisor Dr. Dheerendra Singh, Professor and Head, Department of Computer Science & Engineering,. Without the wise counsel and able guidance, it would have been impossible to complete the thesis in this manner. The constant guidance and encouragement received from Shri. Yogesh Verma, Sci./Engr-SD, GSOAD-RQAG-SRA, SAC-ISRO and Dr. R. Nandakumar, Head, GSOAD-RQAG-SRA, SAC-ISRO, AHMEDABAD has been of great help in carrying out the present work and is acknowledged with reverential thanks. I express gratitude to other faculty members of Department of CSE, for their intellectual support throughout the course of this work.

7. REFERENCES
[1] [3 April, 2013], Cecilia Rodriguez Matrinez “Software quality studies using analytical metric analysis”
[2] [6Dec,2013],DavidRacodon“http://docs.codehaus.org/display/SONAR/metric+definitions”
[3] [April, 2013], Rudiger Lincke, Jonas Lundberg and Weflowe “Comparing Software Metrics Tools”
[4] [2005], QaiserDurrani , “Role of Software Metrics in Software Engineering and Requirements Analysis”
[7] [2, March, 2009], SubhajitDatta , “ Metrics And Techniques To Guide Software Development” Florida State University College Of Arts And Sciences, P.H.D.
[8] [31 March, 2005], Ganesh Kandula, Vinay Kumar Sathrasala, “Product and Management Metrics for Requirements” Master Thesis
[12] [20 Dec., 2001], Vanita Shroff , “Requirements Management Metrics”, Master’s in Software Engineering Department of Electrical and Computer Engineering, University of Calgary, Canada
[13] [2003], James R McCoy, “Requirements use case tool”, Conference on Object Oriented Programming Systems Languages,
[16] [June 1994 ], Marian Jureczko And Diodimis D. Spinellis, “Using Object-Oriented Design Metrics To Predict Software Defects “
[18] [2009], Professor D. Vernon, “Khaliifa University Course Notes “
[20] [Sept 1993], Dewayne E. Perry and Carol S.Steig, “An Empirical Approach to Design Metrics and Judgments”
[25] [2006], Denys Poshyvanyk, “The Conceptual Coupling Metrics for Object-Oriented Systems by Department of Computer Science”, Wayne State University Detroit Michigan

44
[26] [2001], Daniel Rodriguez and Rachel Harrison, “An Overview of Object-Oriented Design Metrics”
[27] [Jan 2007], Wasif Afzal, “Metrics in Software Test Planning and Test Design Processes”, School of Engineering Blekinge Institute of Technology Sweden
[28] [May 2003], Hilda B. Klasky, “A Study Of Software Metrics “ A thesis submitted to the Graduate School-New Brunswick Rutgers, The State University of New Jersey
[29] NachiappanNagappan, “Using Historical In-Process and Product Metrics for Early Estimation of Software Failures”, Microsoft Research
[31] [1999], Elfriede Dustin, Thom Garrett, Bernie Gauf, “Implementing Automated Software Testing”
[33] [Aug 2006], “INFOSYS operational excellence through efficient software testing metrics”.
[34] [2009], “Software Measurements And Metrics: Role In Effective Software Testing”
[35] [Jan 2011] Sheikh Umar Farooq, Department of Computer Sciences, University of Kashmir Srinagar, J&K
[37] [Jan 2007], Wasif Afzal, “Metrics in Software Test Planning and Test Design Processes”, School of Engineering Blekinge Institute of Technology Sweden
[38] [May 2011], Vikas Verma and Sona Malhotra “Applications Of Software Testing Metrics In Constructing Models Of The Software Development Process”.
[40] [2005], Linda Westfall, “12 Steps to Useful Software Metrics”
[43] [Aug 1991], U.S. Department of Transportation Federal Aviation Administration, “Software Quality Metrics Overview”

www.mindlance testing.com.