Metrics Tools For Reengineering Software Projects

Aika Ibraheem Kreedy

Abstract: the ability of detect and predict poor quality of software is of major important to software engineering and reengineering, manger, quality assurance organization. Poor quality software leads to increase development cost and expensive maintenance. With so much attention on exacerbated budgetary constraints, a viable alternative is necessary. Software quality metrics designed for this purpose. Metric measure certain aspect of code or PDL representations, and can collected and use through life cycle. Automated software quality measures are important for easy integration into the software development process. This paper discuss on two metrics: **static analysis tools** and **function point analysis tools,** take examples for each type and discus the different between them.

Index Terms: Software Reengineering, Metrics tools, Static analysis tools, Function point analysis tools.

I. INTRODUCTION

Software analysis generally extracts arbitrary properties of software source code. General or custom analyses of software can be implemented using DMS. Software metrics are a special kind of analysis focused on the structure of the source code. Classic software metrics range in variety from the very simple Source Lines of Code (SLOC) to more complex measures such as Cyclomatic Complexity measurements. Typical metrics report provides details on individual modules and summaries for subsystems. Such metrics are widely used to judge the quality of source code, enabling a software organization to more effectively focus its attention on the lower-quality portions of their portfolio. We will have a look at different metrics tools and we will gain an understanding what kind of different tools are available. Metrics tools can be used for several tasks.

- Estimating projects (i.e. estimating time, effort, cost)
- Managing change of scope
- Measuring productivity
- Communicating functional requirements
- Benchmarking

There are a lot of metrics tools available, of which most are commercial. We have chosen to present the tools that seem to the most popular ones, according to the result of the survey during field study. A Survey on Software Metrics in “Best Practices” Organizations The paper “software metrics best practices – 20.kulik and C. Weber presents a study on software metrics in organizations. The study was done in the 4th quarter of 2001 and the 1st quarter of 2002, and the result described in the paper includes, among other things, commonly used software metrics tools. [KuWe02]

- *Aika Ibraheem Kreedy*
- *Singhania University*
- *Kreedy_aika@yahoo.com*

How the Matrix Works

The Matrix of Change presents a way to capture connections between practices. It graphically displays both reinforcing and interfering organizational processes. Armed with this knowledge, a change agent can use intuitive principles to seek points of leverage and design a smoother transition. Once the broad outlines of the new system and the transition path have been charted, authority can more effectively be decentralized for local implementation and optimization. The Matrix highlights interactions and complementary practices. An example of a collection of critical complements includes the use of flexible machinery, short production runs, and low inventories (Dudley and Lasserre; Milgrom and Roberts). Emphasizing one such practice increases returns to its complementary practices. Likewise, doing less of a given complement reduces returns to its operating dependents. In this example, more flexible machinery draws value from and adds value to shorter production runs. Trouble starts when change agents fail to identify feedback systems that push business units back toward old ways of doing business or when they miss synergy that would strengthen the new and better ways they wish to establish. Ironically, the bottom-up, continuous improvement principles associated with TQM can also be counterproductive - it may be that no single isolated change can improve a process, but a coordinated change can. Incremental change can sometimes be more painful than radical change. Twenty-five years ago, the Swedish government decided to shift from driving on the left side of the road to driving on the right. The scope of the change was enormous. When faced with dramatic change, affected parties often plead for time to adapt. But, imagine the consequences of asking the trucks to drive on the right-hand side during the first month of the transition and then the cars in the second month! Some transitions are smoothest when everyone changes their behavior quickly and at once. Although empowerment and decentralized decisions are popular, this practice can certainly fail if uncoordinated - imagine each driver independently determining the best side of the road for driving. As it turns out, Sweden made the change quickly during the least trafficked hours of night. Once the plan was universally communicated, it was in each individual driver's best interests to comply. The Matrix of Change functions as a four step process. It provides a systematic means to judge those business practices that matter most. It highlights interactions among these practices and possible transition difficulties from one set of practices to another. It encourages various stakeholders to provide feedback on proposed

changes. And, it uses process interactions to provide guidelines on the pace, sequence, feasibility, and location of change.

Background

More than 570 individuals worldwide participated in the study, representing 43 different countries. Over fifty percent of the respondents were a manager or project leader, with almost one fourth of the respondents being at the senior or executive level. Almost half of the organizations of survey respondents have more than 100 software developers. Some of the participating organizations included Boeing, Diebold, EDS, Intel, Lucent, and Nokia.

Metrics tools Used

The most common tool used to capture and analyze software metrics is Microsoft Excel. There is no big difference between usage of metrics tools between "Best practice" organizations and "All others". In general, "Best Practices" organizations are more likely to use industry tools to analyze software metrics regularly. On average, "Best Practices" organizations use 2.1 metrics tools while "All Other" organizations use only 1.4 metrics tools. Table 1 shows the most common metrics tools and their overall usage. In this paper, we present some of the metrics tools listed in the table.

Most Common Metrics Tools
Ms-Excel Miscrosoft (43%)
RationalSuite – Rational Software (17%)
Rational Software (14%)
QSM-SLIM QSM (7%)
McCabe toolset – McCabe & Associates (6%)
Function Piont Workbench Charismatek (5%)
Public Domain tool (5%)
MetricCenter from Distributive Software (3%)

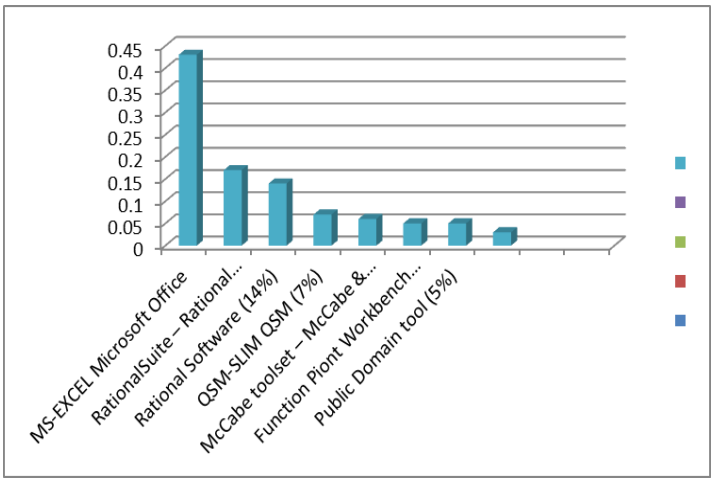

Table & chart (1): Most Common Metrics Tools

Different Metrics Tools

The metrics tools, presented here, are split into two groups: **static analysis tools and function point analysis tools**. There is also a third group consisting of metrics tools that cannot be split into either of these.

Static Analysis Tools

Static Analysis tools analyze automatically the properties of a program without executing it. There are applications for doing the following:

- Collecting information on program structures and dependencies
- Collecting source code metrics
- Checking adherence to programming standards and guidelines

LOC is one the basic static metric that is used to measure the size of code segment. It helps in measuring the cost of project in an effective way. The most commonly used complexity metric before 1990 was cyclomatic [FPMN] complexity that was measured by McCabe. He uses the flow graph and some mathematical equations to compute software complexity. This metric was used in code development risk analysis [10], change risk analysis in maintenance and in test planning. In 1976 McCabe [11] defined the cyclomatic complexity number metric. The metric measures the number of independent paths through a software module. Although cyclomatic complexity is widely used. Examples of use of static analysis metrics tools:

Medical software: The U.S. Food and Drug Administration (FDA) have identified the use of static analysis for medical devices. [ERYGF]

Nuclear software: In the UK the Health and Safety Executive recommends the use of static analysis on Reactor Protection Systems. [DGYTR]

Aviation software (in combination with dynamic analysis) [ADETRY]

A study in 2012 by VDC Research reports that 28.7% of the embedded software engineers surveyed currently use static analysis tools and 39.7% expect to use them within 2 years.

McCabe QA

McCabe QA offers insight into software quality through module-by-module metric calculation. Some of the metrics used by McCabe QA are the following

- McCabe Cyclomatic Complexity
- McCabe Essential Complexity
- Module Design Complexity
- Integration Complexity
- Lines of Code

Metrics measurements are also traced over time to track program improvement. McCabe QA also provides a visual environment for showing graphically the structure of code and the metrics ranking to provide valuable assessment of even large systems. A database, including metrics and flow graphs, is used as a valuable resource for future software changes and upgrades.

LDRA Testbed

LDRA Tesbed is a quality control tool that provides source code analysis and testing facilities for the validating and verification of software applications. It provides analysis which may be applied in the two main testing domains of Static Analysis and Dynamic Analysis. [LDRA] LDRA Testbed's Static Analysis enables a project to ensure that a uniform set of programming standards are enforced, software is properly structured and complexity and other quality attributes are

controlled within a configurable model. The benefits are :

- Better understanding of the system
- Adherence to quality standards
- Identify and eliminate unnecessary code
- Determine the complexity of the system
- Fully automated analysis

Static Analysis is the primary source of information for the automatic documentation of software. It generates the control flow information for each procedure and the inter-procedural links. The interfaces are accurately delineated, the loop structure is exposed and complexity metrics are generated. [LDRASA] Dynamic Analysis explores the semantic of the application under test via test data selection. Control and data flow models constructed from the Static Analysis of the software application are compared with the actual control flow and data flow that are yielded at run time. This enables checks which show errors in either the Static or Dynamic Analysis. The benefits are:

- High quality testing is performed
- Reduce cost and effort of regression testing
- Identifies software defects
- Yields a comprehensive test data set with measureable quality and known test outcomes
- Reduces maintenance costs to minimum
- Identifies unnecessary parts of the system
- Ensures systems are reliable and as error free as possible

Dynamic Analysis is particularly effective for the analysis of software applications which are required to achieve high levels of reliability. [LDRADA]

Function Point Analysis Tools

Function Point Analysis is a reliable method for measuring the size of computer software. In addition to measuring output, function point analysis is very useful in estimating projects, managing change of scope, measuring productivity, and communicating functional requirements.

The Strengths of Function Point Metrics

1. Function point metrics have more measured projects than all other metrics combined.
2. Function points are increasingly used for software contracts.
3. Function Points are easily understood by the non-technical user. This helps communicate sizing information to a user or customer.
4. They can be counted by different people, at different times, to obtain the same measure within a reasonable margin of error.
5. Function Points can be used to size software applications accurately.

The Weaknesses of Function Point Metrics

1. Function point analysis is slow. Counting speeds for function points average perhaps 500 function points per day.
2. Due to the slow speed of function point analysis, function points are almost never used on large systems > 10,000 function points in size.
3. Application size is not constant. During development applications grow at perhaps 2% per calendar month. After development applications continue to grow at perhaps 8% per calendar year. Current counting rules do not include continuous growth.

In today's world software development is a global business. About 60% of Indian companies and more than 80% of Indian outsource companies use function points in order to attract outsource business, with considerable success. As already mentioned Brazil now requires function points for all government outsource contracts. Clearly global competition is a topic of critical interest to all C level executives including CEO's, CFO's, CTO's CIO's, CRO's, and all others. Function point metrics are the best (and only) metric that is effective for very large scale global studies of software productivity and quality. Table 1 shows approximate results for 7 countries.

country	Approximate Software Productivity (FP per Month)	Approximate Defect Potentials in 2013 (Defects per FP)	Approximate Defect Removal Efficiency	Approximate Delivered Defects in 2013 (Defects per FP)
Japan	9.15	4.50	93.50%	0.29
India	11.30	4.90	93.00%	0.34
China	9.15	5.20	86.50%	0.70
Canada	8.85	4.75	91.75%	0.39
US	8.95	4.82	90.15%	0.47
Brazil	9.40	4.75	90.00%	0.48
Iraq	7.95	5.05	82.50%	0.88

Function point Workbench

Function pint Workbench, developed by Charismatek, is not a full software cost estimating tool, but assist in enumerating costs based on function points by allowing efforts and costs to be distributed. The Function Point Workbench tools include full function point sizing capabilities and a repository of projects that have been sized. The function Point Workbench is often used in addition to other forms of software cost estimation tools. The Function Point Workbench provides a tool for the following situations:

- Software definition
- Software estimation
- Outsourcing management
- Benchmarking
- Project control
- Cost negotiation

Embedded within the Function Point Workbench is a documented professional counting methodology based on software modeling. The methodology is applicable throughout the software lifecycle. [FPWo]

Rational Suite

IBM Rational Suite family products provide a complete lifecycle set of integrated solutions. Together they empower a

team to communicate clearly and effectively, share and re-use assets repeatedly, improve team collaboration and optimize individual and team productivity, according to the web page IBM Rational Suite: A Complete, Integrated Lifecycle Solution [IBM]. Rational Suite unifies cross-functional teams and makes individuals and teams more productive. By providing a full complement of development tools, each integrated with other components, individual team members share broader project insight. The testers are alerted when they need to change a test script, a developer is notified when a requirement is reprioritized and a project manager is flagged when the high-priority bug count exceeds a specified limit. What sets Rational Suite apart is the Team Unifying Platform, which includes two types of integrated solutions. Project management solutions helps teams with process guidance, project reporting, and progress tracking. Infrastructure tools represent the foundational solutions necessary to communicate and collaborate throughout the developer's process. The IBM Rational platform for software development is open, extensible, and supported by many products and services. IBM Rational works closely with strategic partners to ensure broad compatibility. An open API ensures maximum flexibility.

In summary the IBM Rational Suite:

- Provides a complete lifecycle solution from requirements to testing
- Offers solutions for enterprise, real-time/embedded and UNIX
- Unifies cross-functional teams via key product integrations and work flow
- Provides visual modeling, code-generation and reverse engineering capabilities
- Improves code quality and application performance with automated testing solutions
- Enhances project predictability with iterative development process
- Includes a proved best practice approach to software development
- Supports team and individual development through an on-line development community

Moose

According to the article “Reengineering Object-Oriented Applications” By S. Duacasse [SDu03], MOOSE is a language independent tool environment to reverse engineer and re-engineer software systems. MOOSE consists of a repository to store models of software systems, and provides facilities to analyze query and navigate them. Models consist of entities representing software artifacts such as classes, methods, etc. MOOSE serves as foundation for re-engineering tools and collaborates with other tools. MOOSE supports all major re-engineering tasks. Its extensibility is inherent to the extensibility of its meta-model. Its design allows for extensions for language-specific features and for tool specific information. Several tools have been built which use the functionalities offered by MOOSE. It is an object-oriented framework and offers as such a great deal of possible interactions with the represented entities. Large systems can be dealt with in a satisfactory way. MOOSE has a layered architecture. there are several ways to import information about software systems. Sources can be directly extracted via the meta-model of Smalltalk (in which MOOSE is implemented) or via the built-in parser. For other source languages MOOSE provides an important interface for CDIF or XMI files based on the FAMIX meta-model. Information is transformed from source code into a source code model. The models are based on the FAMIX meta-model. Every model contains elements representing the software artifacts of the target system. The information in this model can then be analyzed, manipulated and used to trigger code transformations by means of re-factorings. MOOSE provides several services that make the life of a re-engineer easier. Every element in a model is represented by an object, which allows direct interaction of elements, and consequently an easy way to query and navigate a model. MOOSE FINDER is a tool that allows one to compose queries based on different criteria like entity type, properties or relationships, etc. MOOSE EXPLORER proposes a uniform way to represent model information. All entities relationships and newly added entities can be browsed in the same way. The analysis services are mostly implemented as operators that can be run over a model to compute additional information regarding the software elements. For example, metrics can be computed and associated with the software entities, entities can be annotated with additional information such as inferred type information, analysis of the polymorphic invocations, etc. MOOSE has grouping mechanisms, with which it is easy to group several entities into one group entity, which is treated from then on as an entity itself. This is useful when a re-engineer wants to reduce the amount of information by looking at the subject system from a higher level of abstraction. The Re-factoring Engine implements language-independent re-factoring. The functionality which is provided by MOOSE is to be used by tools. This is represented by the top layer. Tools can use the repository and services of MOOSE and use the Tools Integration Framework to find each other and integrate.

QSM-SLIM

QSM’s Software Lifecycle Management (SLIM) tools support decision making at each stage of the software lifecycle: estimating, tracking, and benchmarking and metrics analysis. [QSM] QSM-SLIM consist of four tools:

- SLIM-Estimate helps you estimate the time, effort, and cost required to satisfy a given set of software requirements and determine the best strategy for designing and implementing your software project.
- SLIM-Control has the statistical process control techniques you need to assess the status of your project (compare the project plan against project actuals and generate a forecast to completion).
- SLIM-Metrics works with SLIM-DataManager to preserve project history, assess competitive position, identify bottlenecks, quantify the benefits of process improvement, and defend future project estimates.
- EstimateExpress is QSM’s software project estimating tool for organizations with smaller estimating requirements. EstimageExpress calculates the cost, schedule, reliability, and resources for large and small software projects while providing the ability to negotiable and plan multiple project scenarios. Backed by QSM and our worldwide industry database of thousands of software projects, Express is a perfect choice for software development shops that don’t need an enterprise solution.

For several years after the discovery of LOC is one the basic static metric that is used to measure the size of code segment. Problems IBM used LOC for coding in the true language for coding the application, but used basic assembly language to derive ratios for non-coding work. This was an awkward method and explains why IBM invested several million dollars in developing both function point metrics and a formal parametric estimation tool that could handle estimates for applications in all programming languages.For that we asking this question how do we measure economic productivity? The standard definition for economic productivity is “goods or services produced per unit of labor or expense.” Let us consider the results for the sum of the coding and non-coding effort using both function points per staff month and lines of code per staff month. Table 2 shows both values:

Table 2: Function Points versus LOC per Month for Calculating Economic Productivity Rates for selected languages.

Languages	Function Pts. per Month	LOC per Month
Machine language	1.45	927.54
Basic Assembly	2.70	864.86
JCL	3.69	815.29
Macro Assembly	3.80	810.13
HTML	4.76	761.90
C	5.62	719.10
Basic (interpreted)	8.77	561.40
Quick Basic	9.01	549.36
C++	9.68	516.13
Java	9.68	516.13
PHP	9.68	516.13

CONCLUSION

Software reengineering practitioners { architects, developers, managers { must be able to rely on scientific results. Especially research results on software quality reengineering and metrics should be reliable. They are used during Re-engineering, to take early measures if parts of a system deviate from the given quality speculations, or during maintenance, to predict effort for maintenance activities and to identify parts of a system needing attention. In order to provide these reliable scientific results, quite some research has been conducted in the area of software metrics. In this paper we discuss chosen metrics tools that seem to the most popular ones according to the result of some studies, the metrics tools presented in this paper are split into groups: First, **Static analysis tools** which analyze automatically the properties of program without executing it. Second group **Function point analysis** tools are a reliable method for measuring the size of software and there is also third group consisting of metrics tools that cannot be split in to either of these and we take one examples for each type of metrics tool which we mention above to discuss it in detail. In addition we mention the use of FPA in 7 countries in all over the world and at end we make comparative study between difference between static analysis tools and function point , for that we take LOC tools as type of static analysis to compare it with Function point tools from economic productivity rate factor in Table[2], which show clearly how Function point tools more economic then LOC, as comparative study to static analysis tools and Function point tools according to specific factors:

1. Cost 2.accurate 3.complexity 4.purpose 5.effective 6.speed

Factor	FPA	Static Analysis
➢ Cost ➢ Accurate ➢ purpose ➢ complexity ➢ speed	➢ Less ➢ More ➢ Sizing, study software productivity, quality, cost, risk and economic value. ➢ Less ➢ slow	➢ More ➢ Less ➢ Primary measurement for Collecting information on program structures & dependencies, Source code metrics& Checking adherence to programming standards and guidelines. ➢ more ➢ fast

REFERENCE

[1] [DGYTR] Computer based safety systems - technical guidance for assessing software aspects of digital computer based protection systems, http://www.hse.gov.uk/nuclear/operational/tech_asst_guides/tast046.pdf

[2] [ERYGF] FDA (2010-09-08). "Infusion Pump Software Safety Research at FDA". Food and Drug Administration. Retrieved 2010-09-09

[3] [KuWe02] Peter kulik, Catherine Weber, Software Metrics Best Practices -2001, March 2002, http://visualbasic.ittoolbox.com/pub/PK043002.pdf

[4] [SDu03] S. Ducasse, Reengineering Object-Oriented Applications, University of Bern, Institute Fur Informatik and Angewandte Mathematik, 2003, http://www.iam.unibe.ch/~scg/Archive/Papers/Duca01cHab.pdf

[5] [FPWo] Function Point WORBENCH, http://www.charismatek.com.au/_au/_public1/html1/fpw_brouchure.htm

[6] [IBM] IBM Rational Suite: A Complete, Integrated Lifecycle Solution, http://www.pts.com/rsuite.cfm

[7] [QSM] QSM Products Overview, http://www.qsm.com/products.html

[8] [LDRA] LDRA: Software Development & Testing with LDRA TestBed, http://www.ldra.co.uk/pages/testbed.asp

[9] [LDRASA] LDRA: Static Analysis with LDRA TestBed, http://www.ldra.co.uk/pages/staticanalysis.asp

[10] [LDRADA] LDRA: Dynamic Analysis with LDRA TestBed, http://www.ldra.co.uk/pages/dynamicanalysis.asp.

[11] [ADETRY] Position Paper CAST-9. Considerations for Evaluating Safety Engineering Approaches to Software Assurance // FAA, Certification Authorities Software Team (CAST), January, 2002: "Verification. A combination of both static and dynamic analyses should be specified by the applicant/developer and applied to the software.

[12] [FDSWJI] H F Li, W K Cheung “An Empirical Study of Software Metrics” Software Engineering IEEE Transactions on (1987) Volume: SE-13, Issue: 6, Pages: 697-708

[13] [YWQRP] N E Fenton “Software Metrics” Conference Proceedings of on the future of Software engineering ICSE 00(2000) Volume: 8, Issue: 2, Publisher: ACM Press

[14] [FPMN] [10] Manik Sharma, Dr. Gurdev Singh, “Analysis of static and Dynamic Metrics for Productivity and Time Complexity”, IJCA, Volume 30– No.1, September 2011.