Research Paper Information Technology

Proposed Software Metrics for Software Development for Contribution to Maintainability

*Kirti Mathur ** Amber Jain

* International Institute of Professional Studies, D. A. University, Indore

** International Institute of Professional Studies, D. A. University, Indore

ABSTRACT

Software Metric is the measure specific property of software. Metrics mainly help in the maintainability of the software by helping to diagnose the problem area, bug fixing and meeting new requirements. Maintainability of software is the degree to which it can be understood and corrected. Software developers often spend at least 70% of their project budget and time on the software maintenance. This paper discusses how to select metrics that can be helpful to developers for maintainability.

Keywords : Software, metrics, maintainability, modifiability, deterioration

1. Introduction

Software metrics are a quantitative measure of the degree to which a system, component or process possesses a given attribute[5]. It takes more than one and different metrics for each person in development team to understand, evaluate or control a software product, process, service or project[10].

Maintainability means the ability of software to be maintained, correct and adopt new changes in itself.

Software metrics can be used by developers for software maintainability. Not all metrics have the same discriminatory power when it comes to predicting the quality of a software object[2].

2. LITERATURE REVIEW

2.1 Software Metrics:

It is a quantitative measure of the degree to which a system, component or process possesses a given attribute[5]. Metrics can help us **understand, evaluate** and **control** our software products, processes, services and **predict** attributes of software entities[14].

2.2 Way to find proper software metrics:

Software maintainability is a difficult factor to quantify. Software metrics can provide maintainability statistics required by management as well as engineers for making technical decisions. A proper approach find software metrics involves:

2.2.1 Identify the customer who is going to use the metric:

Customer type	Matrix form required
Functional and Project Managers:	- Schedules - Budget
Software Developers/ Programmers	- Time spent per task - Inspection of data including defects - Root cause of defects
Software Testers	- Test cases - Planned/executed/pa-ssed - Problem reports from Testing - Test coverage
Configuration management Specialists	- Lines of code - Data changed

Table-1: Metrics used by customers

2.2.2 Select one or more measurable goals:

Organizational level goals: This includes[8]:

- high-level strategic goals[8] (like low cost provider)
- maintaining high level of customer satisfaction
- meeting projected revenue and profit margin target.

Project level goals: This includes:

- goals that emphasize project management
- control issues or project level requirements and objectives

2.2.3 Ask questions:

This is to ensure that each goal is being obtained. For example, if our goal was to ship only defect-free software, questions might be:

- Is the software product adequately tested?
- How many defects are still undetected?
- *Are all known defects corrected?*

2.2.4 Select metrics:

Remember software metrics don't solve problems, People solve problems[10]. Software metrics act as indicators, so people can make more informed decisions and intelligent choices.

2.2.5 Find the standard definitions:

Find the standard definitions for the entities and their measured attributes. Terms like defect, problem report, size, project, quality, maintainability, and user-friendliness are ambiguous have different meanings for different individuals. For example: defect report, problem report, incident report or fault report may be used by various organizations to mean the same thing, but unfortunately they may also refer to different entities.

Differing interpretations of terminology is the biggest barriers to understanding[4]. Unfortunately, there is little standardization in the industry of the definitions for most software attributes. The suggested approach is to adopt standard definitions within your organization and then apply them consistently.

2.2.6 Choose a measurement function for the metrics:

Measurement function defines how we are going to calculate the metric. **Base measures** (or metric primitives) are meas-

ured directly and their measurement function typically consists of a single variable. **Derived measures** *are modeled using mathematical combinations of base or derived measures. Examples of measures include:*

- *number of lines of code reviewed during an inspection*
- *hours spent preparing for an inspection meeting.*
- *inspection's preparation rate (number of lines of code reviewed divided by the number of preparation hours).*

2.3 Software maintainability:
Software doesn't change, but factors such as bugs, new ideas/features, organizational priorities, laws, project sponsors, users, new operating systems and hardware changes force it to change. Software systems are built under high pressure to meet deadlines with emphasis on performance, reliability, and usability[1]. Maintainability is the ease of maintaining a software product such that it can be helpful to isolate and correct defects or their causes*[6].*

Maintenance Type	Description
Preventive	It is the act of being proactive towards the problem before it occurs.
Adaptive	Adaptive maintenance refers to the act of making the software adapt the new environment, which is different from the earlier one.
Corrective	It sometimes also refers to as 'repair' and brings back the software to the working condition
Predictive	Predictive maintenance is the act of maintaining software when software requires a change

Figure-1: Maintainence types

2.4 People responsible for the maintainability:
Everyone in development team is responsible for the maintainability:
- Project manager/Functional manager: Responsible for controlling the project size, resources required, budgeting and scheduling activities.
- Developer/Programmer: Responsible for the actual development, designing and coding of software
- Tester: Responsible for testing that the software works as expected meeting the requirements that guided its design and development.

2.5 Software Maintainability Characteristics:
- Effect of maintainability: Software deteriorates with aging. Due to careless changes and number of times the maintainability has been implemented, at some point software can't incorporate new changes.
- Maintainability never stops: *Changes are inevitable.*

2.6 Type of metrics for software maintainability
2.6.1 Metrics for Software Management:

Productivity	· Number of lines of code/modules/classes/ deliverables developed in unit time or per resource.
Quality:	· Project complexity · Portfolio complexity · Degree of client/executive management satisfaction
Deliverables	· Ratio between achieved and planned deliverables. · Number of reworks because of no co-ordinances between specifications and results.
Costs:	Statistics regarding: · different costs categories · project portfolio value · resources usage and costs · resource loading and distribution.
Risks:	· Number of identified risks · Number of raised risks · Number of avoided risks.

Table-2: Metrics for managers

2.6.2 Metrics for Developers:
- lines of code written
- user tasks completed
- bugs fixed
- tests written
- tests passing first time
- bugs found
- code churn vs. new code (i.e. "write first time" vs "rewritten repeatedly")
- Percentage of time in IDE vs. debugging
- Percentage of time in IDE vs. non-work applications
- code performance (against some arbitrary or customer-specified benchmark)

The best metrics tend to be combinations (e.g. average of bugs found per line of code written) rather than a single measure.

2.6.3 Metrics for Testers:

Customer satisfacti-on index	· Number of system enhancement and maintenance fix requests per year · User friendliness in training new users and customer service · Number of product recalls or fix releases and reruns
Delivered defect quantities	· Requirements defect · Design defect · Code defect · Documentation defects · Defect introduced by fixes, etc.
Delivered defect quantities and Responsi-veness to users	· Time for minor vs. major enhancements · Actual vs. planned elapsed time
Product volatility	· Ratio of maintenance fixes vs. enhancement requests
Defect ratios	· Defects found after product delivery: · per function point · per LOC
Defect removal efficiency	· Number of post-release defects (reported by users) · Ratio of defects found internally prior to release as a percentage of all defects · All defects including defects found developers and users in the first year
Complex-ity of delivered product	· McCabe's cyclomatic complexity counts across the system · Halstead's measure · Card's design complexity measures · Predicted defects and maintenance costs, based on complexity measures
Test coverage	· Breadth of functional coverage · Percentage of paths, branches or conditions that were tested · Percentage by criticality level · Ratio of number of detected faults to the number of predicted faults.
Cost of defects	· Business losses per defect that occurs during operation · Business interruption costs; costs of workarounds · Lost sales and lost goodwill · Litigation costs resulting from defects · Annual maintenance and operating cost (per function point)

Costs of quality activities	Cost of: · reviews, inspections, preventive measures · test planning preparation · test execution, defect tracking, version change control · diagnostics, debugging , fixing · tools and tool support · test case library maintenance · testing & QA education associated with the product · monitoring and oversight by QA organization
Re-work	Re-worked: · effort · LOC · software components
Reliability	· Availability · Mean time between failure (MTBF). · Man time to repair (MTTR) · Reliability ratio (MTBF / MTTR) · Number of product recalls or fix releases and production re-runs as a ratio of production runs

Table-3: Metrics for testers

2.6.4 Metrics for customers:

Reliab-ility	Company's ability to perform the promised service dependably and accurately.	· Meeting customer Specifications · Products/Services/ Modules works Right first time · Consistency Performance/Availability · Accuracy and completeness of Service
Assur-ance	Employees' ability to convey trust and confidence and their knowledge, competence and courtesy	· Are Materials provided (Training manuals/ Broachers/Presentations up-to date) · Provided honest and trustworthy services · Customer data safety assured
Tangi-bles	Physical facilities, equipment and appearances that impress the customer	· Ease of support access · Demonstrate customers and issues understanding
Empa-thy	Level of caring, individuali-zed attention, access, communica-tion and understand-ing perceived by customer	· Ease of use of products/ services · Easy to understand written materials · Satisfaction with IT-Infrastructure
Respo-nsiven-ess	Willingness to help clients and provide prompt service	· Speed/willingness of Response · Commitment given and met

Table-4: Metrics for Customers

3. Relation between software metrics and maintainability

Software metrics can estimate only the programmers' opinion of maintainability. Choosing appropriate metrics for measuring the maintainability depends on product's nature and the programming language used[4]. Using a metrics framework allows programmers to locate modules and routines with a low level of maintainability. Metrics assist programmers to inspect their code and make the necessary corrections and improvements during the implementation phase. Software metrics are aimed for the improved estimation of readability, clearness, sufficiency of comments and simplicity of code[3].

4. Recommendation for keeping software maintainable

When the software is first being developed:

- Code readability should be a primary goal.
- Setup automated testing.
- Use version control software.
- Software should be easy to understand, make changes, test, operate, deploy.

5. CONCLUSION AND FUTURE WORK

Software metrics play an important role in maintainability of software. Well formulated metrics can help organizations to improve software quality.

- Software maintainability is not only restricted to the metrics evaluation but in future also can be combined with Configuration Management System of the Software Evolution.
- This metrics evaluation can be useful for object-oriented software.

REFERENCES

1. How much information do software metrics contain, Yossi Gil, Maayan Goldstein, Dany Moshkovich, 2009 | 2. A systematic review of software maintainability prediction and metrics 2009- Mehwish Riaz, Emilia Mendes, Ewan Tempero, The University of Auckland | 3. Security metrics for software systems 2009, Ju An Wang, GA Hao Wang, Minzhe Guo, Min Xia | 4. Designing Maintainability in Software Engineering: a Quantified Approach, 2008 Tom Gilb | 5. An empirical analysis of the impact of software development problem factors on software maintainability 2008 - Jie-Cherng Chen, Sun-Jen Huang, National Taiwan University of Science and Technology | 6. Software Maintenance: Similarity and Inclusion of Rules in Knowledge Extraction 2006 Marek Reformat, Alberta Univ., Edmonton, Alta. Aashima Kapoor ; Nicolino J. Pizzi | 7. Experience measuring maintainability in software product lines 2006 - Gentzane Aldekoa, Salvador Trujillo, Goiuria Sagardui, Oscar Díaz | 8. An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks 2006, Ko, A.J. Human-Comput. Interaction Inst., Carnegie Mellon Univ., Pittsburgh, PA , Myers, B.A. ; Coblenz, M.J. ;Aung, H.H. | 9. Rodrigo Vivanco and Nicolino Pizzi, 2004 Finding effective software metrics by using parallel generic algorithm 2004 | 10. An Integrated Measure of Software Maintainability, In Proceedings of Annual Reliability and Maintainability Symposium, IEEE, 2002. Aggarwal K. K., Singh Y., and Chhabra J. K. | 11. Carolyn Seaman 2002, The Information Gathering Strategies of Software Maintainers. | 12. Rikard Land 2002, Software Deterioration and Maintainability – A Model Proposal. | 13. Dimitris Stavrinoudis, Michalis Xenos, Dimitris Christodoulakis 1999, Relation between software metrics and maintainability. | 14. A Comparative Survey of Software Quality | Metrics, Kirti Mathur, Amber Jain, Paripex, April 2013, |