

Software Quality Metrics

BY SAMER ZEIN, PHD

Introduction



“You cannot control what you cannot measure”, Tom Demarco (1982)

Software Metrics Definitions:

- *“A quantitative measure of the degree to which an item possesses a given quality attribute.”*
- *“A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute.”*

Introduction..2

Software metrics should be included in software development, in order to :

- Control of SW development & maintenance
- Support of decision making
- Initiation of corrective actions

Analysis of SW metrics could show the results of certain interventions such as :

- New development tools,
- Changed procedures
- etc

Introduction..3

However Metrics as a quality assurance tool has not, unfortunately, been applied at an adequate level in the software industry.

Nor have they provided benefits at the anticipated levels.

Only a small portion of software development organizations apply software quality metrics systematically, and few of these report successful use of the results of their efforts.

Objectives of SW Metrics

Frame 21.1 Main objectives of software quality metrics

(1) **To facilitate management control as well as planning and execution of the appropriate managerial interventions.** Achievement of this objective is based on calculation of metrics regarding:

- Deviations of actual functional (quality) performance from planned performance
- Deviations of actual timetable and budget performance from planned performance.

(2) **To identify situations that require or enable development or maintenance process improvement in the form of preventive or corrective actions introduced throughout the organization.** Achievement of this objective is based on:

- Accumulation of metrics information regarding the performance of teams, units, etc.

It is all about Comparison

Comparison provides the practical basis for management's application of metrics and for SQA improvement in general.

The metrics are used for the **comparison** of **performance** data with *indicators, and quantitative values such as:*

- *Defined SW quality standards*
- *Quality **targets** set for organizations*
- *Previous **year** quality achievements*
- *Previous **Project** quality achievements*
- *Average **quality levels** achieved by other similar **teams***

Frame 21.2

Software quality metrics – requirements

General requirements

Explanation

Relevant

Related to an attribute of substantial importance

Valid

Measures the required attribute

Reliable

Produces similar results when applied under similar conditions

Comprehensive

Applicable to a large variety of implementations and situations

Mutually exclusive

Does not measure attributes measured by other metrics

Operative requirements

Explanation

Easy and simple

The implementation of the metrics data collection is simple and is performed with minimal resources

Does not require independent data collection

Metrics data collection is integrated with other project data collection systems: employee attendance, wages, cost accounting, etc. In addition to its efficiency aspects, this requirement contributes to coordination of all information systems serving the organization

Measures of System Size

KLOC – this classic metric measures the size of software by thousands of code lines. As the number of code lines required for programming a given task differs substantially with each programming tool, this measure is specific to the programming language or development tool used. Application of metrics that include KLOC is limited to software systems developed using the same programming language or development tool

Measures of System Size...2

Function points – a measure of the development resources (human resources) required to develop a program, based on the functionality specified for the software system

SW Process Quality Metrics: Error Density Metrics



Calculation of error density metrics involves two measures:

- (1) ***software volume***, : use of number of lines of code
- and (2) ***errors counted***:
 - ***Number of errors***
 - **Weighted number of errors**



Example:

Error severity class	Relative weight
Low severity	1
Medium severity	3
High severity	9



Error severity class <i>a</i>	Calculation of NCE (number of errors) <i>b</i>	Calculation of WCE	
		Relative weight <i>c</i>	Weighted errors $D = b \times c$
Low severity	42	1	42
Medium severity	17	3	51
High severity	11	9	99
Total	70	—	192
NCE	70 	—	—
WCE	—	—	192 

Code	Name	Calculation formula
CED	Code Error Density	$CED = \frac{NCE}{KLOC}$
DED	Development Error Density	$DED = \frac{NDE}{KLOC}$
WCED	Weighted Code Error Density	$WCED = \frac{WCE}{KLOC}$
WDED	Weighted Development Error Density	$WDED = \frac{WDE}{KLOC}$
WCEF	Weighted Code Errors per Function point	$WCEF = \frac{WCE}{NFP}$
WDEF	Weighted Development Errors per Function point	$WDEF = \frac{WDE}{NFP}$

Key:

- NCE = number of code errors detected in the software code by code inspections and testing. Data for this measure are culled from code inspection and testing reports.
- KLOC = thousands of lines of code.
- NDE = total number of development (design and code) errors detected in the software development process. Data for this measure are found in the various design and code reviews and testing reports conducted.
- WCE = weighted code errors detected. The sources of data for this metric are the same as those for NCE.
- WDE = total weighted development (design and code) errors detected in development of the software. The sources of data for this metric are the same as those for NDE.
- NFP = number of function points required for development of the software. Sources for the number of function points are professional surveys of the relevant software.

Example 2. This example follows Example 1 and introduces the factor of weighted measures so as to demonstrate the implications of their use. A software development department applies two alternative metrics for calculation of code error density: CED and WCED. The unit determined the following indicators for unacceptable software quality: $CED > 2$ and $WCED > 4$. For our calculations we apply the three classes of quality and their relative weights and the code error summary for the Atlantis project mentioned in Example 1. The software system size is 40 KLOC. Calculation of the two metrics resulted in the following:

Measures and metrics	Calculation of CED (Code Error Density)	Calculation of WCED (Weighted Code Error Density)
NCE	70	—
WCE	—	192
KLOC	40	40
CED (NCE/KLOC)	1.75 	—
WCED (WCE/KLOC)	—	4.8 

Error Severity Metrics

Code	Name	Calculation formula
ASCE	Average Severity of Code Errors	$ASCE = \frac{WCE}{NCE}$
ASDE	Average Severity of Development Errors	$ASDE = \frac{WDE}{NDE}$

Software Process Timetable Metrics

Software process timetable metrics may be based on accounts of success (completion of milestones per schedule) in addition to failure events (non-completion per schedule).

An alternative approach calculates the average delay in completion of milestones.

Table 21.3: Software process timetable metrics

Code	Name	Calculation formula
TTO	Time Table Observance	$TTO = \frac{MSOT}{MS}$
ADMC	Average Delay of Milestone Completion	$ADMC = \frac{TCDAM}{MS}$

Key:

- MSOT = milestones completed on time.
- MS = total number of milestones.
- TCDAM = total Completion Delays (days, weeks, etc.) for All Milestones. To calculate this measure, delays reported for all relevant milestones are summed up. Milestones completed on time or before schedule are considered “0” delays. Some professionals refer to completion of milestones before schedule as “minus” delays. These are considered to balance the effect of accounted-for delays (we might call the latter “plus” delays). In these cases, the value of the ADMC may be lower than the value obtained according to the metric originally suggested.

Error Removal Effectiveness Metrics

Software developers can measure the effectiveness of error removal by the software quality assurance system after a period of **regular operation** (usually 6 or 12 months) of the system.

The metrics combine the **error records of the development stage** with the **failures records** compiled during the first year (or any defined period) of regular operation.

Error Removal Effectiveness

Code	Name	Calculation formula
DERE	Development Errors Removal Effectiveness	$DERE = \frac{NDE}{NDE + NYF}$
DWERE	Development Weighted Errors Removal Effectiveness	$DWERE = \frac{WDE}{WDE + WYF}$

Key:

- NYF = number of software failures detected during a year of maintenance service.
- WYF = weighted number of software failures detected during a year of maintenance service.

Software Process Productivity Metrics

deal with a project's human resources productivity as well as "indirect" metrics that focus on the extent of software reuse.

Software reuse substantially affects productivity and effectiveness.

Code	Name	Calculation formula
DevP	Development Productivity	$DevP = \frac{DevH}{KLOC}$
FDevP	Function point Development Productivity	$FDevP = \frac{DevH}{NFP}$
CRe	Code Reuse	$CRe = \frac{ReKLOC}{KLOC}$
DocRe	Documentation Reuse	$DocRe = \frac{ReDoc}{NDoc}$

Key:

- DevH = total working hours invested in the development of the software system.
- ReKLOC = number of thousands of reused lines of code.
- ReDoc = number of reused pages of documentation.
- NDoc = number of pages of documentation.

Limitations of Software Quality Metrics

Application of quality metrics is strewn with obstacles. These can be grouped as follows:

- Budget constraints in allocating the necessary resources (manpower, funds, etc.) for development of a quality metrics system and its regular application.
- Human factors, especially opposition of employees to evaluation of their activities.
- Uncertainty regarding the data's validity, rooted in partial and biased reporting.