The Survey of Metrices on Software Quality Assurance and Reuse

Ashwin Tomar
MCA Computer Science Dept.
Siddhant Inst of Comp Applicat
Under Pune Univ. MH, INDIA

V. M. Thakare
P.G. Computer Science Dept.
Amravati University
Amravati, MH, INDIA

ABSTRACT
In this paper, we surveyed Software Quality Assurance metrics useful in the different phase of the Object-Oriented Software Development Life Cycle. Recent work in the field is summarized and an outlook for software metrics in quality assurance and reuse is provided. Metrics are used by the software industry for analyzing, designing & development, implementation and maintenance of software. The practice of applying software metrics to a software process and to a software product brings knowledge about the status of the process and / or product of software in regards to the goals to achieve. In this paper, we have presented Software Quality assurance and reuse metrics which has not been emphasized.

General Terms
Software quality assurance i.e. SQA, Software reuse, Metrics

Keywords
Characteristics, Attributes, Audit Metric (AM)

1. INTRODUCTION
"Quality" may be defined as "conformance to requirements". Such "conformance" means, most generally, that the product meets the needs of the user and satisfies stated performance criteria. Quality is "Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected from all professionally developed software" [1]. Quality is multidimensional aspect. Quality can be viewed from different view i.e. from supplier view and customer view. From supplier view quality is doing the right thing, in right way, on right time and from customer view quality is delivering the right product, satisfying the customer needs, expectation, treating customer with respect, integrity etc. Quality comprises all characteristics and significant features of a product or an activity which relate to satisfying of given requirements. Software should have quality means it should have low defect potential, high defect removal efficiency, stable requirements, high user satisfying ratings. Objective is to optimes quality and productivity by controlling various factors affecting them.

There are different characteristics which software should have like maintainability, flexibility, testability, efficiency, correctness, reliability, integrity, usability, interoperability, portability etc. These are suggested in McCall's, Dromey's, FURPS, ISO 9126, and SQuaRE model [1]. They have used the word i.e. Quality factor. Quality factor are factors which influence quality. The Product if developed is below standard level i.e. low quality, it means that it has more errors, faults and bugs. Such software below quality standard will cause heavy loss of revenue, life, business problems like insurance hazards, banking hazards, taxes calculation hazards, hospital and health hazards. Quality is competitive and challenging issue, it is must for survival.

Quality Assurance is process of developing the product [1]. It is the process of ensuring that a quality work product in every phase of software development. It deals with how quality is achieved; it is estimating required level of quality. It is the process of verifying or confirming whether the software product and services meet the customer expectation. It is a process driven and preventive approach. Quality assurance process has to control to imbibe, accumulate quality into a product. It depends on many factors like planning, standards, rules, legal procedures, documentation, guidelines, technology, authority, approvals, environment, risk, size, report, reuse, virtue and ethics [6]. Our aim is to optimize quality and productivity. Software quality assurance defines the standards (established by Govt. or any organization) for the software products developed in its organizational unit. SQA specify and implement tools and aids for assessing software product quality [2]. The tools may be as simple as checklists or sophisticated as ones that automatically count the number of unique instruction types in a program, the number of conditional jumps in it, or other such elements that may have a bearing on software quality. In short SQA is a set of planned and systematic set of activities that provide adequate confidence about establishing the requirements properly and assures that the products or services conform to specified requirements. It is preventive approach which prevents the faults from occurring by providing rules and methods. It is the task conducted on process (not product). It evaluates process to produce the products.

Software measurement is concerned with deriving a numeric value for an attribute of a software product or process. Metric is a quantitative measure of the degree to which a system component or process possesses a given attribute [1]. It is quantifiable measurement of software product, process or project that is directly observed, calculated or predicted. Metrics are indicator. They are computed from measures [14]. Metrics form the basis for planning, evaluation and controlling. They are useful to monitor requirements, predict development resources, track development progress and understand maintenance costs. The advantages of software metrics are to compare study of various design methodology, to analysis, compare & study programing languages with respect to their characteristics, in evaluating capabilities and productivity of people, in preparation of quality specification, in getting idea of complexity, guiding managers in SDLC.

2. CLASSIFICATION & METRICS

Authors have classified metrics in different ways like Core/ non-core metric, Direct/ indirect metrics, Process/ product [4], Primitive/computed metrics. They have classified it from commercial perspective[13] like technical metric, defect metric, end user satisfaction, warranty metrics, reputation metrics etc. Direct metrics are easy to establish, actual, tangible and quantifiable. They are used to evaluate software.

2.1 SQA Metrics

Software metrics can be classified into three categories: **Project metrics** are those that describe the project characteristics and execution. E.g. include the number of software developers, the staffing pattern. **Process metrics** are management metrics [4] which are used for improving the software development and maintenance process. e.g. include the effectiveness of defect removal during development, the pattern of testing defect arrival, and the response time of the process. **Product metrics** are quality metrics [13] and are used to measure the properties of software. like mean time to failure, defect density customer problems, customer satisfaction.

Other way of classifying is **primitive metrics or computed metrics** [14]. Primitive metrics are those that can be directly observed. Program size (in LOC), number of defects observed in unit testing, or total development time for the project. Computed metrics are those that cannot be directly observed but are computed in some manner from other metrics. Computed metrics are combinations of other metric values and often more valuable in understanding or evaluating the software process than are simple metrics. Examples of computed metrics are those commonly used for productivity, such as LOC produced per person-month (LOC/person-month), or for product quality, such as the number of defects per thousand lines of code (defects/KLOC).

McConnell has classified metrics in different way [1] as:-
Size Metrics: Total LOC, Total comment lines, Total data declarations, Total blank lines.
Productivity Metrics: Work-hours spent on the project, Work-hours spent on each routine changed, Dollars spent on project, Dollars spent per line of code, Dollars spent per defect.
Defect Tracking Metrics: Severity of each defect, Location of each defect, Way in which each defect is corrected, Person responsible for each defect, Number of lines affected by each defect correction, Work hours spent in correcting each defect, Average time required to find a defect, Average time required to fix a defect, Attempts made to correct each defect, Number of new errors resulting from defect correction.
Overall Quality Metrics: Total number of defects, Number of defects in each routine, Average defects per thousand lines of code, Mean time between failures, Compiler-detected errors.
Maintainability Metrics : Number of parameters passed to each routine, Number of local variables used by each routine, Number of routines called by each routine, Number of decision points in each routine, Control-flow complexity in each routine, Lines of code in each routine, Lines of comments in each routine, Number of data declarations in each routine, Number of blank lines in each routine, Number of jumps in each routine, Number of input, output statements in each routine. **Code Coverage Metrics**: Percentage of methods under test, Percentage of classes under test.

"952" Metrics from Software Quality Assurance (SQA)
There is list of 952 metrics for SQA downloaded from Internet which is listed in this paper. There are fourteen metrics associated with it shown as - (14) which are listed

1.0 Software Quality Assurance Metrics
1.1 Software Audit Metrics
1.1.1 Software Activity Audit Metrics (14)
1.1.2 Software Product Audit Metrics (14)
1.1.3 Software Walkthrough Audit Metrics ([14)
1.1.4 Software Inspection Audit Metrics (14)
1.1.5 Software CMM® Audit Metrics (14)
1.2 Documentation Audit Metrics
1.2.1 Minimum Documentation Audit Metrics
1.2.1.1 Software Requirements Document (SRD) Audit Metrics (14)
1.2.1.2 Software Architecture Document (SAD) Audit Metrics (14)
1.2.1.3 Software Verification and Validation Plan (SVVP) Document Audit Metrics (14)
1.2.1.4 Software Verification and Validation Report (SVVR) Document Audit Metrics (14)
1.2.1.5 Software User Documentation Description (UDD) Audit Metrics (14)
1.2.1.6 Software Configuration Management Plan (SCMP) Audit Metrics (14)
1.2.2 Other Documentation Audit Metric
1.2.2.1 Software Project Plan (SPP) Audit Metrics (14)
1.2.2.2 System Requirements Specification (SRD) Audit Metrics (14)
1.2.2.3 System Architecture and Requirements Allocation Description (SARAD) Audit Metrics (14)
1.2.2.4 Database Design Description (DDD) Audit Metrics (14)
1.2.2.5 Software Interface Design Description (SIDD) Audit Metrics (14)
1.2.2.6 Test or Validation Plan (TVPL) Audit Metrics (14)
1.2.2.7 Software Design Description (SDD) Audit Metrics (14)
1.2.2.8 Test or Validation Procedures (TVPR) Audit Metrics (14)
1.2.2.9 Test or Validation Results Report (TVRR) Audit Metrics (14)
1.2.2.10 Software Integration Plan (SOIP) Audit Metrics (14)
1.2.2.11 Software Integration Audit Report (SIAR) Audit Metrics (14)
1.2.2.12 Software Installation Plan (SIP) Audit Metrics (14)
1.3 Review and Audit Metrics
1.3.1 Minimum Review and Audit Metrics
1.3.1.1 Software Requirements Review (SRR) Metrics (14)
1.3.1.2 Software Preliminary Design Review (SPDR) Metrics (14)
1.3.1.3 Software Critical Design Review (SCDR) Metrics (14)
1.3.1.4 Software Verification and Validation Plan Review (SVVPR) Metrics (14)
1.3.1.5 Function Configuration Audit (FCA) Metrics (14)
1.3.1.6 Physical Configuration Audit (PCA) Metrics (14)
1.3.1.7 In-Process Audit (IPA) Metrics (14)
1.3.1.8 Managerial Review (MR) Metrics (14)
1.3.1.9 Software Configuration Management Plan Review (SCMPR) Metrics (14)
1.3.1.10 Post Mortem Review (PMR) Metrics (14)
1.3.2 Other Review and Audit Metrics
1.3.2.1 System/Subsystem Requirements Review (SSRR) Metrics (14)
1.3.2.2 System/Subsystem Design Review (SSDR) Metrics (14)

1.3.2.3 Software Test Readiness Review (SOTRR)Metrics (14)
1.3.2.4 Software Test Results Review (SOTRER) Metrics (14)
1.3.2.5 System Test Readiness Review (SYTRR) Metrics (14)
1.3.2.6 System Test Results Review (SYTRER) Metrics (14)
1.3.2.7 Software Usability Review (SUR) Metrics (14)
1.3.2.8 Software Maintenance Review (SMR) Metrics (14)
1.4 Test Audit Metrics
1.4.1 Software Unit Testing Audit Metrics (14)
1.4.2 Software Integration Audit Metrics (14)
1.4.3 Software Qualification Testing Audit Metrics (14)
1.4.4 System Integration Audit Metrics(14)
1.4.5 System Qualification Testing Audit Metrics (14)
1.5 Code Control Metrics (14)
1.6 Media Control Metrics (14)
1.7 Supplier Control Metrics (14)
1.8 Risk Management Metrics (14)
1.9 Software CMM Key Process Area (KPA) Audit Metrics (Audit Metric = AM)
1.9.1 Software CMM Level 2 Key Process Area (KPA) AM
1.9.1.1 Requirements Management (RM) Audit Metrics (14)
1.9.1.2 Software Project Planning (SPP) Audit Metrics (14)
1.9.1.3 Software Project Tracking and Oversight (SPT&O) Audit Metrics (14)
1.9.1.4 Software Subcontract Management (SSM) AM (14)
1.9.1.5 Software Quality Assurance (SQA) Audit Metrics (14)
1.9.1.6 Software Configuration Management (SCM) AM (14)
1.9.2 Software CMM Level 3 Key Process Area (KPA) Audit Metrices (14)
1.9.2.1 Organization Process Focus (OPF) Audit Metrics (14)
1.9.2.2 Organization Process Definition (OPD) A M (14)
1.9.2.3 Training Program (TP) Audit Metrics (14)
1.9.2.4 Integrated Software Management (ISM) AM (14)
1.9.2.5 Software Product Engineering (SPE) AM (14)
1.9.2.6 Intergroup Coordination (IC) Audit Metrics (14)
1.9.2.7 Peer Reviews (PR) Audit Metrics (14)
1.9.3 Software CMM Level 4 Key Process Area (KPA) Audit Metrics (14)
1.9.3.1 Quantitative Process Management (QPM) A M (14)
1.9.3.2 Software Quality Management (SQM) Audit Metrics (14)
1.9.4 Software CMM Level 5 Key Process Area (KPA) Audit Metrics (14)
1.9.4.1 Defect Prevention (DP) Audit Metrics (14)
1.9.4.2 Technology Change Management (TCM) Audit Metrices (14)
1.9.4.3 Process Change Management (PCM) Audit Metrics (14)

The above list of common (14) metrics associated with it are

1 Number of Scheduled type of Audits
2 Number of Actual type of Audits
3 Ratio of Scheduled to Actual type of Audits
4 Effort per type of Audit
5 Number of Deviations per type of Audit
6 Number of Deviations Open type of Audit
7 Number of Deviations Closed type of Audit
8 Number of Actual Corrections type of Audit
9 Number of Deferred Corrections type of Audit
10 Ratio of Actual/Deferred Corrections type Audit
11 Ratio of Actual Corrections to Effort type of Audit
12 Number of Major Corrections type of Audit
13 Number of Minor Corrections per type of Audit
14 Ratio of Major Corrections to Effort per type of Audit

Metrics are also classified under following head as [12]:

Traditional Metrics
McCabe Cyclometric Complexity (CC)
Source Lines of Code (SLOC)
Comment Percentage (CP)

C.K. Metrics [12]
Weighted Method per Class (WMC)
Depth of Inheritance Tree (DIT)
Number of children (NOC)
Coupling between objects (CBO)
Response for a Class (RFC)
Lack of Cohesion in Methods (LCOM)

MOOD Metrics [12]

Encapsulation	Method Hiding Factor (MHF)
	Attribute Hiding Factor (AHF)
Inheritance	Method Inheritance Factor (MIF)
	Attribute Inheritance Factor (AIF)
Polymorphism	Polymorphism Factor (POF)
Coupling	Coupling Factor (COF**)**

Quality Metrics [1]
Mean time to failure (MTTF)
Mean time to repair (MTTR)
Mean time between failures (MTBF)
MTBF=MTTF+MTTR
Probability of failure on demand (POFOD)
Rate of failure occurrence (ROCOF)
Availability AVAIL = MTTF/ (MTTF+MTTR)*100%
Lines of Code (KLOC)
Defect Removal Efficiency (DRE = E/ (E+D)

Different Researcher have classified metrics with example under different categories like Size metrics (LOC, Token counts i.e. operator, operands, function counts), Control Flow metrics (McCabe's Cyclometric Metrices), Information Flow metrics (FAN-IN, FAN-OUT), Testing Metrices (Test coverage, defect acceptance, defect aging, defect density, review efficiency, defect severity index, test effectiveness etc.), Coupling, Inheritance (no of children, parents, ancestor, descendants, methods overridden), Cohesion Metrices (Tight and loose)[3], Design metrics and many other metises. These matrices are directly and indirectly associated with software quality assurance.

Metrices are classified under four categories on basis of software development phase. Requirement & Analysis (Specific, Complete, Validated requirements), Design (Information flow, The Bang Metric, Functions Points, Cyclometric complexity), implementation (Estimation of number of defects, Lines of code, Product metrics of Halstead i.e. Program vocabulary, length, volume), Testing & Maintenance (defect metrics, Software reliability), Effort & cost metric (Empirical metrics, Statistical model, Halstead metric) [5].

Product Quality Metrices are Customer satisfaction index, defect ratio, defect removal efficiency, responsiveness to users, product volatility, test coverage, cost of defects, cost of quality activities, reliability, re-work, complexity of delivered product which can be listed under product quality metrics[14].Software reuse is systematic and managerial process. It can be used for large scale and in building single system.

2.2 Reuse Metrics

Software reuse is process of creating software system from existing software rather than building them from scratch. Software reuse improves quality and productivity. Measurement is considered as one of the success factor in favoring reuse. Metrics plays an important role in quality assurance & reuse which decides whether a component should be used or not.

Reuse models and metrics can be categorized into six types [16]:
1. reuse cost-benefits models
2. maturity assessment
3. amount of reuse
4 .failure modes
5 .reusability
6 .reuse library metrics (reuse repository)

This list of classifying reuse metrics was downloaded from internet [7]. Some metrics relate directly to a measurement, some require calculation of formulas relying on direct measurements.

Management Metric Samples
Number of Individual Rewards Issued
Number of Team Honors Issued
Number of reward incentives released
Number of honor incentives released

Qualifier Metric Samples
Number of RC requests not filled
Effort Statistics for evaluation of an artifact

Supplier Metric Samples
Number of artifacts submitted by this Supplier
Number of RCs generated from artifacts submitted by this Supplier
Number of a particular Supplier's RCs reused by others
Number of error reports filed for a particular Supplier's RCs

Aggregate Supplier Metrics
Total number of Suppliers
Total number of submitted artifacts
Artifacts submitted / Supplier
Number of accepted RCs
Number of reused RCs
Number of Error Reports
Number of open error reports

Individual Component User Metrics - for a particular user:
Number of RC searches
Number of RCs checked-out
Number of RCs reused
Number of User's own RCs reused by self
Number of RCs suggested
Number of RCs suggested resulting in RC being built
Number of searches not resulting in candidate RCs
Number of searches not resulting in reuse of one of the candidate RCs
Participation as User and Supplier = Number of artifacts submitted + Number of RCs checked-out + Number of RCs suggested + Number of searches
Positive Participation = Number of RCs proposed by this User (as a Supplier) accepted + Number of RCs reused by this User + Number of RCs suggested resulting in an RC being built + Number of own (as Supplier) RCs reused

Reusable Component (RC) Metrics for a particular RC
RC size (other quality and descriptive metrics defined elsewhere)
Original artifact effort
Artifact extraction effort
Artifact acceptance rework effort
Artifact categorization effort
RC creation effort
RC modification effort
Number of hits
Number of check-outs
Number of open check-outs
Number of reuses
Number of indirect reuses
Estimated RC non-reuse cost = Original artifact effort ***** (Number of reuses + Number of indirect reuses)
Number of error reports
Number of open error reports
Check-out to reuse ratio = Number of check-outs/Number of reuses

Qualification Level
Aggregate RC modification effort (sum, mean, standard deviation, average efforts)
To date Effort saved by reuse

Check-out Metrics for a particular RC
Check-out result (open, curious inquiry, reused, related component reused, not-reused-why)
Check-out time open

Reuse History Metrics for a particular RC
Modifications required
Modification effort
Application integration effort (for reuse)
Predicted development effort (if no reuse)
Percent effort saved with reuse

Error Metrics for a particular RC
(Usual collection of error data)

Repository Content Metrics
Number of RCs in the Repository
Number of versioned components
Total number of check-outs
Total number of open check-outs
Number of RCs reused (per time period)
Total number of reuses
Total number of reuses with no modifications
Number of artifacts accepted without rework
Number of artifacts accepted with rework
Number of open error reports
Number of error reports filed
Number of RCs with reported errors
Number of RCs with open error reports
Mean error reports per <size measure> for each type of RC
Library check-out to reuse ratio (goodness of cataloging scheme)
Effort saved by reuse for repository
Reuse effort for repository
Estimated non-reuse cost for repository
Collection creation effort

Reuse metrics are also classified as below [10]:

Economics Oriented Reuse Metrics (EORM) - are related to return on investment (ROI) models and the economic impacts of reuse based on a set of observable data [10].
a) Reuse Cost Avoidance (RCA)
b) Reuse Value Added (RVA)
c) Organizational or Project level ROI

Software Structure Oriented Metrics (SORM) - are concerned on what is being reused and how it is being reused from a strictly technical standpoint.
a) **Reuse level (RP)**: Ratio of the number of reused lines of code to the total number of lines of code.
b) **Reuse Level (RL):** Ratio of the number of reused items to the total number of items.

c) **Reuse Frequency (RF)**: Ratio of the references to reused items to the total number of references.
d)**Reuse size & Frequency (RSF*)*:** Similar to Reuse Frequency, but also considers the size of items in the number of lines of code.
e)**Reuse Ratio (RR):** Similar to Reuse percent, but also considers partially changed items as reused.
f) **Reuse Density:** Ratio of the number of reused parts to the total number of lines of code.
Reuse Repository Metrics (RRM) - assess the overall performance of the convergence point of the reuse activities in an organization.
a) Cost to find adequate reusable parts
b) Quality of contained reusable parts
c) Number of successful reuse cases of contained parts.
d) Availability of the repository'
e) Total value of a reuse repository

Metrics for measuring the amount of generosity [8] included in the code. These are:
Function Template Factor (FTF): It is the ratio of Number of Functions using Function Templates to total Number of Function.
Class Template Factor (CTF): It is defined as ratio of Number of Classes using Class Templates and Number of Classes.

3. CONCLUSION & FUTURE WORK

In this paper we gave emphasis on metrics which are least used. We surveyed metrics on quality assurance and reuse and presented a summary of it. Most of metrics are collected from papers and internet. Quality assurance is the process of building quality in software. It is defect preventive process. There are many factors on quality assurance which are optimized to build quality in software. Software reuse is the use of existing software artifacts in the development of other software artifacts with the goal of improving productivity and quality among other factors.

Metrics provides basis for analyzing, measuring, comparing, evaluating all characteristics of product, process (quality assurance and reuse) and people. Metrics quantifies the characteristics. It serves as indicator which gives value to characteristics. This paper will help researcher involved in developing components (reuse technology) by standardizing process.

4. REFERENCES

[1] www.onestoptesting.com, www.stickyminds.com www.softwaretestinggenius.com, www.chetanaforum.com, www.softwaretestinginterviewfaqs.com

[2] J. E. Gaffney, Jr, 1981," Metrics In Software Quality Assurance," ACM '81, November 9-11.

[3] Amjan Shaik, C. R. K. Reddy, Bala Manda, Prakashini C, Deepthi. K, Metrics for Object Oriented Design Software Systems A Survey, Journal of Emerging Trends in Engineering and Applied Sciences 2010.

[4] Gurdev Singh, Dilbag Singh, Vikram Singh, " A study of Software Metrics", IJCEM, vol 11, Jan 2011

[5] Dr.B.R.Sastry, Saradhi Vijaya,"Impact of Software Metrics on Object Oriented Software Development life cycle", IJOEST, Vol 2(2), 2010, 67-76.

[6] Tomar Ashwin, Thakare. V.M, "Identification and listing of factors affecting SQA", International Conference on Computer Science and Information Technology, IRNET, 14 July 2012.

[7] www.Toolbox.com

[8] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra, "Proceedings of the 2005 Third ACIS Int'l Conference on Software Engineering Research, Management and Applications", 2005 IEEE.

[9] Mrinal Singh Rawat, Arpita Mittal, Sanjay Kumar Dubey,"Survey on Impact of Software Metrics on Software Quality", International Journal of Advanced Computer Science and Applications, Vol. 3, No. 1, 2012.

[10] In Distributed Systems, Jorge Cláudio Cordeiro Pires Mascena, Eduardo Santana de Almeida, Sílvio Romero de Lemos Meira,"A Comparative Study on Software Reuse Metrics and Economic Models from a Traceability Perspective", 2005 IEEE.

[12] Arora Deepak, Khanna Pooja, Tripathi A, Sharam S, "Software Quality Estimation through Object Oriented Design Metrics", IJCSNS, Vole 11, N0-4, April 2011.

[13] Stephen H. Kan Tavel, Book on Software quality Engineering.

[14] Farooq Sheikh U, Quadri S M K, Ahmad, "Software Measurement and Metrics: Role in Effective Software Testing", IJEST, Vole 3, No 1, Jan 2011.

[15] Stephen H. Kan Tavel, Book on Software quality engineering.

[16] Frakes, William and Carol, Terry,"Software Reuse: Metrics and Models" ACM Computing Surveys 28 (2), pp. 415-435, 1996.

[17] Xunmei G U, Jun SHI, Reuse Metrics for Object-Oriented Method, 2010 IEEE.

[18] Ashwin Tomar, V. M .Thakare, "The Study of Software Reuse and Models", IJCA Proceedings on National Conference on Innovative Paradigms in Engineering and Technology (NCIPET 2012).