ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 9(44), DOI: 10.17485/ijst/2016/v9i44/86988, November 2016*

A Survey on Prioritization of Software Quality Attributes

R. Senthilkumar* and T. Arunkumar

SCOPE, VIT University, Vellore - 632014, Tamil Nadu, India;
senvpm73@gmail.com, rsenthilkumar@vit.ac.in

Abstract

Objective: The objective of the survey is to understand the need of software quality metrics in providing better software product. Without knowledge about the quality dependencies, conflicts, incurred costs, and technical feasibility it is very difficult to minimize the cost involved in software development. **Statistical Analysis:** The Quantitative feedback for the prioritization and cost/benefit considerations for quality requirements is highly recommended. The distinction between the utilization of software metrics and software quality confirmation is necessary to quantify the strength software product against vulnerabilities. The prioritization of software quality attributes is demonstrated to influence the software developers for providing the quality software product. The prioritization is established, product and process metrics. **Findings**: Various distinctive metrics identifying with upkeep are depicted. This will be trailed by brief exchanges about programmed gathering of software metrics information, use of gathered information, and expenses of software metrics. This survey has been formed with remarkably the quality components reasonableness and steady quality likewise, the quality model versatile quality identity a primary need. It is intended for my future investigation eagerness of ensuring nature of programming by means of mechanized data collection of the product quality measurements. **Application**: The benefits of an instrument, or a gadget set, to be conveyed, will preferably be to manage the bolster costs in a relationship in a more orchestrated and proficient way.

Keywords: Prioritization, Process Metrics, Product Metrics, Project Metrics, Quality Attributes, Software Quality Metrics

1. Introduction

Estimation is finished by metrics. Three parameters are measured: process estimation through process metrics, product estimation through product metrics, and project estimation through project metrics. Process metrics survey the adequacy and nature of programming process, decide development of the process, exertion required in the process, viability of imperfection evacuation amid advancement. Product metrics is the estimation of work product delivered amid distinctive periods of programming improvement. Project metrics delineate the project attributes and their execution.

1.1 Process Metrics

To enhance any process, it is important to gauge its predefined traits, add to an arrangement of significant metrics in light of these properties, and afterward utilize these metrics to get markers with a specific end goal to infer a procedure for process change.

Utilizing programming process metrics, programming architects can evaluate the effectiveness of the product process that is performed utilizing the process as a structure. Process is set at the focal point of the triangle joining three variables (product, individuals, and innovation), which have an imperative impact on programming

**Author for correspondence*

quality and association execution. The aptitude and inspiration of the general population, the unpredictability of the product and the level of innovation utilized as a part of the product advancement have an imperative impact on the quality and group execution. The process triangle exists inside of the circle of ecological conditions, which incorporates advancement environment, business conditions, and client/client qualities.

To quantify the productivity and adequacy of the product process, an arrangement of metrics is defined in view of the results got from the process. These results are recorded underneath.

1. Number of mistakes found before the product discharge
2. Deformity recognized and reported by the client after conveyance of the product
3. Time spent in settling blunders
4. Work products conveyed
5. Human exertion utilized
6. Time used
7. Adjustment to plan
8. Hold up time
9. Number of agreement adjustments
10. Assessed cost contrasted with real cost.

Note that process metrics can likewise be inferred utilizing the attributes of a specific programming building action. For instance, an association might quantify the exertion and time spent by considering the client interface plan. It is watched that process metrics are of two sorts, to be specific, private and open. Private Metrics are private to the individual and serve as a pointer just for the predetermined individual(s). Deformity rates by a product module and imperfection blunders by an individual are cases of private process metrics. Note that some process metrics are open to all colleagues yet private to the project. These incorporate mistakes identified while performing formal specialized surveys and absconds reported about different capacities incorporated into the product.

Open metrics incorporate data that was private to both people and groups. Project-level imperfection rates, exertion and related information are gathered, broke down and surveyed with a specific end goal to get markers that assistance in enhancing the authoritative process execution.

1.2 Product Metrics

In programming advancement process, a working product is created toward the end of each effective stage. Every product can be measured at any phase of its improvement. Metrics are created for these products with the goal that they can show whether a product is created by client prerequisites. In the event that a product does not meet client prerequisites, then the essential moves are made in the individual stage. Product metrics help programming designer to identify and rectify potential issues before they bring about disastrous imperfections. Likewise, product metrics survey the interior product ascribes with a specific end goal to know the proficiency of the accompanying.

- Examination, outline, and code model
- Power of experiments
- General nature of the product a work in progress
- Different metrics detailed for products in the improvement process are recorded beneath.
- Metrics for analysis model: These location different parts of the analysis model, for example, framework usefulness, framework size, etc.
- Metrics for design model: These permit programming specialists to evaluate the nature of design and incorporate compositional design metrics, part level design metrics, et cetera.
- Metrics for source code: These evaluate source code multifaceted nature, viability, and different attributes.
- Metrics for testing: These design proficient and successful experiments furthermore assess the viability of testing.
- Metrics for support: These survey the solidness of the product.
- Metrics for the Analysis Model

There are just a couple of metrics that have been proposed for the analysis model. On the other hand, it

is conceivable to utilize metrics for project estimation in the connection of the analysis model. These metrics are utilized to look at the analysis model with the goal of foreseeing the span of the resultant framework. Size goes about as a marker of expanded coding, reconciliation, and testing exertion; at times it likewise goes about as a pointer of many-sided quality included in the product design. Capacity point and lines of code are the regularly utilized systems for size estimation.

1.3 Project Metrics

Project metrics empower the project supervisors to survey current projects, track potential dangers, distinguish issue regions, modify work process, and assess the project group's capacity to control the nature of work products. Note that project metrics are utilized for strategic purposes instead of key purposes utilized by the process metrics. Project metrics fill two needs. One, they minimize the improvement plan by making important modification keeping in mind the end goal to evade postpones and reduce potential dangers and issues. Two, these metrics are utilized to survey the product quality all the time and alter the specialized issues if required. As the nature of the project enhances, the quantity of mistakes and surrenders are diminished, which thusly prompts a decline in the general expense of a product project.

Frequently, the first utilization of project metrics happens amid estimation. Here, metrics gathered from past projects go about as a base from which exertion and time gauges for the present project are figured. As the project continues, unique assessments of exertion and time are contrasted and the new measures of exertion and time. This correlation offers the project supervisor to screen (some assistance with supervising) and control the advancement of the project. As the process of improvement continues, project metrics are utilized to track the blunders identified amid every advancement stage. For instance, as programming develops from design to coding, project metrics are gathered to survey nature of the design and acquire markers that thus influence the methodology decided for coding and testing. Likewise, project metrics are utilized to gauge production rate, which is measured as far as models created, capacity focuses, and conveyed lines of code.

2. Software Quality Attributes

To begin with in a nutshell, know what is Quality? Quality can be characterized in distinctive way. Quality definition might vary from individual to individual. However, at long last there ought to be a few guidelines. So Quality can be characterized as

- These are some quality definitions from alternate point of view. Presently let's perceive in what manner one can quantify some quality traits of item or application.
- The following are the factors that are utilized to gauge programming advancement quality. Every scribe can be utilized to quantify the item performance. These characteristics can be utilized for Quality certification and in addition Quality control. Quality Assurance exercises are oriented towards counteractive action of presentation of imperfections and Quality control exercises are gone for identifying deformities in items and administrations.
- Unwavering quality: Measure if item is sufficiently solid to maintain in any condition. Should provide the reliably results. Item dependability is measured as far as working of venture under diverse working environment and distinctive conditions.
- Viability: Distinctive variants of the item ought to be anything but difficult to keep up. For improvement it is ought to be anything but difficult to add code to existing framework, ought to be anything but difficult to overhaul for new components and new advances time to time. Upkeep ought to be financially savvy and simple. Framework be anything but difficult to keep up and correcting absconds or rolling out an improvement in the product.
- Convenience: This can be measured as far as convenience. Application sought to be easy to use but difficult to learn. Route ought to be basic. The framework must be simple to use for information arrangement, operation, and translation of yield. Give steady client interface guidelines or traditions with our other often utilized frameworks

- Portability: This can be measured as far as costing issues identified with porting, Technical issues identified with porting, Behavioral issues identified with porting.
- Correctness: Application ought to be correct as far as its usefulness, figuring utilized inside and the route ought to be correct. This implies application ought to stick to utilitarian necessities.
- Productivity: To Major framework quality trait. Measured regarding time required to finish any assignment given to the framework. For instance, framework ought to use processor limit, circle space and memory productively. In the event that framework is utilizing all the accessible assets then client will get corrupted performance falling flat the framework for productivity. In the event that framework is not proficient then it cannot be utilized as a part of constant applications.
- Security: Uprightness accompanies security. Framework trustworthiness or security ought to be adequate to anticipate unauthorized access to framework capacities, averting information misfortune, guarantee that the product is shielded from infection contamination, and ensuring the protection of information went into the framework.
- Testability: Framework ought to be anything but difficult to test and find deformities. In the event that required ought to be anything but difficult to partition in diverse modules for testing.
- Adaptability: Should be sufficiently adaptable to alter. Versatile to different items with which it needs connection. But it is difficult to understand the interface with the other standard segments.
- Reusability: Programming reuse is a decent cost productive and efficient advancement way. Distinctive code libraries classes ought to be sufficiently nonexclusive to utilize effectively in diverse application modules. Isolating application into diverse modules with the goal that modules can be reused over the application.
- Interoperability: Interoperability of one framework to another ought to be simple for item to trade information or administrations with different frameworks. Diverse framework modules ought to work on distinctive working framework platforms, diverse databases and conventions conditions.

Applying above quality characteristics measures we can figure out if framework meets the prerequisites of value or not. As indicated over every one of these characteristics are connected on QA and QC prepare so that analyzer or client additionally can discover nature of use.

3. Prioritize the Software Quality Attributes

From undertaking to extend, the components of value that are important will fluctuate colossally. Inserted frameworks might have basic proficiency prerequisites given constraints of space and control; however, ease of use contemplations may be totally immaterial. Then again, while handheld applications might see comparative productivity necessities to those of conventional installed frameworks, ease of use will be an important thought for selection and separation from the opposition.

A compelling method for prioritizing the scope of value credits is to handle the issue in two stages. This makes the process proficient, yet at the same time shields us from missing any important components. For these prioritization steps, it is important that all partner groups are included in the talk. A typical botch in necessities examination is to choose prerequisites by representing a partner that we don't enough speak to, and it is exceptionally uncommon for an investigator to have the capacity to authoritatively settle on choices for the expansive scope of categories. The primary stage is to consider each of the traits thus, and to figure out whether there is any present information that would make that trait completely in or out of extension for this task. As shown in the Table 1, a handheld item to be assembled for a particular working environment, with no desire of changes to this environment, will not have to consider portability issues. This same item, however, due to constraints of memory or preparing limit, might need to conclusively manage effectiveness issues at this stage. Not these characteristics can be at the same time streamlined, and for sure, some of the regions of value are inconsistent with each other

Table 1. Prioritization of software quality attributes

Attribute	Score	Availability	Usability	Maintainability	Reusability	Portability
Availability	0		high	high	high	high
Usability	4			less	less	less
Maintainability	3				less	less
Reusability	2					
Portability	1					

(for instance, a framework that is construct to suit high proficiency will intrinsically be less viable,[1–3] gives an astounding lattice demonstrating these tradeoffs).

For the most part, it is find that at any rate half of the considerable numbers of qualities are completely in or out of degree, while it might be using this progression a shorter rundown, this outcome in a little time funds contrasted with the expanded risk of neglecting basic prerequisite. By slashing the rundown into equal parts (which is regular), It diminish the quantity of examinations by a factor of four. We find that numerous correlations are very straightforward, and some will create critical discourse. Once more, we can't know this ahead of time, or without sensible representation from a scope of partners. When we are done this, we have limited the huge rundown we began with, utilizing two methodologies, and are left with a subset of value qualities that are positioned in general importance for our particular undertaking. We have done nothing to really deliver testable prerequisites as of right now, yet we are currently prepared to do as such, and our limited center will guarantee that we fabricate the suitable necessities. This separating and prioritization sets aside little time in an engaged session.

4. Translate into Quantifiable Criteria

The terms or credits that have used to this point are not effortlessly confined in a quantifiable manner. To indicate "the framework should be easy to understand" presents to us no closer to testable necessities, and a sensible way to deal with getting to more fitting terms is lost from large portions of today's conspicuous wellsprings of information on prerequisites.

This basic step is really one that has been around for decades, and is more frequently found in books identified with programming quality or quality designing than to books concentrated on programming necessities. What we have to do here is perform an interpretation, from the scene of value that has guaranteed we are legitimately secured, to a corresponding arrangement of criteria that will permit us to more effectively determine our sought results in measured, testable terms. There are huge numbers of these testable criteria that we can look over. Table 2 demonstrates a curtailed rundown of criteria that guide to quality characteristics. A more thorough rundown

Table 2. Mapping of software quality attributes to criteria

	Reliability	Robustness	Availablility	Flexibililty	Usability	safety	Testability	portability
Error handling		Y			Y	y		
Hazard Analysis		y				y		
Inline code Use							y	
Modularity	y			y			y	y
MTBF	y		y					
MTTR			y					
Simplicity	y			y		y	y	y
Training				y				

of very nearly 50 criteria is caught in a spreadsheet that maps these connections.

It is likely utilized a large portion of these as particular measures for undertakings before. It can be extending from indicated GUI norms and reaction times that can a support ease of use, Mean Time to Between Failures and Mean Time to Repair that together measure accessibility prerequisites, and numerous others. From your experience, you will probably have the capacity to add others to the rundown also.

This table has developed from various distinctive sources. While looking into current necessities based books for quality prerequisites[3,4], we get a couple of samples of terms that are utilized for particular quality characteristics, (for example, reaction times for convenience), however no place close to a rundown that covers all criteria, and no examination of the need to perform the mapping from ascribes to criteria distinguished here. To be sure, in one mainstream message, [5] the author essentially regrets "the fluffy thought of ease of use" and gives no further guidance on the best way to determine quantifiable quality prerequisites. Rather, we have to jump into Quality based references, for example, [6–8] for a more finish depiction of this methodology. Pressman demonstrates the relationship between the properties and criteria, however does not give a reasonable way from the wide credits to concrete quantifiable prerequisites articulations. There are several key focuses here. By and large, we have to recognize more than one of these criteria to give satisfactory scope of any one part of value (else, we would just determine our quality necessities in the original terms we had already distinguished). Also, the majority of the criteria we can utilize will at any rate in part fulfill more than one of these parts of value in the meantime (determined error taking care of systems can support power, ease of use and wellbeing, for instance). In the event that a paradigm in any event mostly fulfills two or more of the properties we consider as important, (for example, Modularity in Table 2), it is a more alluring and productive model to utilize.

5. Summary

This paper is a review of the software quality metrics found in the software designing writing. Meanings of the terms quality and software quality by diverse creators were exhibited. The relationship between software quality confirmation and utilization of metrics were talked about. The metrics displayed were grouped by sorts. The sorts were: established, product furthermore, process metrics. Definitions on product and process situated metrics by a few creators were likewise exhibited. The utilization of the metrics was in no time talked about and in addition the expenses of software metrics. This review has been composed with uncommonly the quality elements practicality and unwavering quality what's more, the quality model many-sided quality personality a main priority. It is designed for my future exploration enthusiasm of guaranteeing quality of software via computerized information accumulation of the software quality metrics. The advantages of an instrument, or a device set, to be delivered, will ideally be to deal with the support costs in an association in a more arranged and efficient way.

6. References

1. Leffingwell D, Widrig D. Managing software requirements: A use case approach. 2nd Edition, Addison Wesley: USA; 2003. p. 503.
2. Amalarethinam DIG, Beena TLC. Level based task prioritization scheduling for small workflows in cloud environment. Indian Journal of Science and Technology. 2015 Dec; 8(33):1–7.
3. Call JM, Richards P, Walters G. Factors in software quality. National Technical Information Service. 1977; 1:1–168.
4. Wiegers K. Software requirements. 2nd Edition, Microsoft Press: USA; 2003.
5. Chamoli S, Tenne G, Bhatia S. Analysing software metrics for accurate dynamic defect prediction models. Indian Journal of Science and Technology. 2015 Feb; 8(S4):96–100.
6. Lauesen S. Software requirements: styles and techniques. Addison Wesley: USA; 2001.
7. Galin D. Software quality assurance: From theory to implementation. Pearson/Addison-Wesley: USA; 2004.
8. Rawat MS, Mittal A, Dubey SK. Survey on impact of software metrics on software quality. International Journal of Advanced Computer Sciences Applications. 2012; 3(1):137–41.
9. Rajasekaran S, Thangavelu A, Dhavachelvan P, Gunasekaran G. Query base k-DRM for software security. Indian Journal of Science and Technology. 2015 Aug; 8(17):1–5.