Software Quality Metrics

Formal Writing Assignment

CS390 – Software Engineering

Derek Burrows

October 20, 2008
Software quality metrics provide a way for software engineers to measure the degree of accuracy of a piece of software in an objective and empirical manner. Traditional quality metrics have included things such as errors per line of code. However, in today's business model the need for better measures of software quality is ever increasing. According to Dr. Bill Curtis, there are four main problems that drastically affect the new business model [1]. The first of which is application malfunction, or more specifically the damage caused by a piece of software when it does not behave in the expected manner. Due to the new business model and its dependence on computers to do most of the work, if the software doesn't calculate someone's bill correctly, not only are is the company potentially losing money due to the incorrect billing, but if the mistake is found, the company must pay someone to correct the mistake and bill the customer correctly. The second problem is business agility. Business agility is the ability of a company, and its software, to adapt to a change, such as the acquisition of another smaller company. If the current software cannot be modified, or added to, then whole new software must be written, costing the business much. The third problem is supplier dependence, or in other words the dependency upon applications that were created by some external party, such as a program produced by a contractor. Without a measure of the quality of, say, an outsourced piece of software, a business has no way of knowing whether the cost of outsourcing was actually worth it, or better quality software could be developed internally. The last problem plaguing the modern business model is application ownership, or the cost associated with maintaining a piece of software. If a piece of software's quality is not maintained, then when new modifications need to be made, the amount of time that programmers must spend to understand how it works skyrockets and the more likely that they will break the existing code when making the changes, again costing the business quite a lot. One can alleviate or avoid all of these problems by increasing the quality of the software through the examination of better metrics based upon quality characteristics that account for these problems. This, in turn, will greatly benefit the business employing the said software.

The question that arises, however, is what exactly are the characteristics of a quality piece of
software? What characteristics should one create metrics from? According to Dr. Bill Curtis, there are five software quality characteristics that address all of the modern business model's problems, transferability, changeability, robustness, performance, and security [1]. Transferability is the aspect of a piece of software that describes how easily new programmers are able to understand how the software's code works and how to make changes or additions to the code. Increasing a software's transferability will not only reduce the amount of time required by a new programmer to learn the code, thereby greatly helping with application ownership costs, but also reduce the chance of introducing a new error when modifications are made, helping avoid application malfunction. Changeability reflects how easily a piece of software can be modified or new components added to it. This has a direct effect upon business agility in an obvious way. The more changeable the software is, the easier it will be able to be modified to account for the changes in the business. Changeability also lowers application ownership costs, by reducing the amount of time needed to actually modify the code. Robustness is the characteristic of software to be able to be modified and still remain stable and operate as expected. A robust program will deter application malfunctions and lower application ownership costs, as well. Performance refers to the speed at which a piece of software can perform its function. It has a large role in the productivity of the business, obviously a slow piece of software will become a bottleneck and hamper the ability of workers to perform their jobs. The last software quality characteristic is security. Security is the ability of software to prevent use or access by unauthorized persons, including but not limited to hackers or perhaps lower level employees who are not supposed to have access. Security has an effect upon application malfunction in the sense that it helps to prevent code changes or data loss by mistake or even perhaps by malicious persons. By restricting access from unauthorized sources, one lessens the possibility of an incompetent or ill-intended individual gaining access and changing or deleting code or data. By creating and studying metrics based upon these characteristics one can increase the quality of their software and avoid the above mentioned problems in the new business model altogether.

The next question that comes about is how do we go about implementing and deploying the
collection and analysis of these new metrics, based upon the five software quality characteristics, to the
software development process? As with most endeavors in life, things do not usually happen as they are
expected to. Changing the software development process in its entirety may lead to unforeseen problems
and challenges. For example, the collection and analysis of metrics can place extra workloads on
programmers and management, causing deadline issues or a decrease in code quality due to time
constraint. According to Dr. Bill Curtis, the best idea is to begin collecting and analyzing metrics in one
smaller portion of the larger application [1]. Based upon the results from this small portion, essentially
what worked and what did not in the implementation of the new method, one can learn from their
mistakes and decide what changes need to be made to the process in order to make the global
implementation more effective and reduce unforeseen side-effects. Once the changes have been made,
one can begin to deploy the process to several other small areas of application and repeat the learning and
refining of the process. Soon the collection and analysis of the metrics will be universal throughout the
application, and the ill-effects should be mostly non-existent. The larger picture of the implementation
and deployment of these metrics, though, must include their utilization by the programmers themselves.
Without the active improvement of the code by the developers via the analysis of these metrics, the
metrics serve no purpose whatsoever. It will need to be reinforced that these metrics are the key to
betterment of their product, not an evaluation of their personal worth as a programmer. The rate at which
these new metrics are implemented, refined, deployed,, analyzed, and utilized directly influences how
soon the business will benefit.

The last question that one should ask is while these metrics are being deployed and used, where in
the business and by whom will the metrics be collected, analyzed, and utilized? Who is responsible for
the use and application of these metrics? According to Dr. Bill Curtis, the responsibilities for the
collection and use of software quality metrics can be broken down by function into four separate groups,
governance, management, evaluation, and improvement, and be divided amongst several levels of the
business [1]. The first group, governance, includes the functions that should be performed by the leaders
of the business, i.e., the executives or those who report directly to the executives. Governance includes decisions about investment and retirement of a software product based on the software quality metrics, managing agreements with external software sources based upon the metrics collected about their software, and analyzing the benefits to the business from the increased software quality. The second group of functions, management, falls upon the application and project managers to facilitate. These functions include predicting the costs, risks, and benefits that will affect the business, creating milestones, goals, and expectations for software quality, and reinforcing individual/team excellence, all based upon the collected software quality metrics. The next group is evaluation, which lies with the actual developers. This is arguably the most important set of functions because it involves the actual collection and application of software metrics during development or maintenance. Not only are these functions extremely important because this is when the programmers actively collect metrics about the quality of the software but because this is exactly when the software's quality is increased via decisions, based on the metrics, about the code itself. Along with collecting metrics about their own code, it is the developers' responsibility to provide metrics about the code produced from external sources, such as contractors or other vendors. The last group is improvement, which also happens to be the programmers'/developers' responsibility. In essence, this last group involves the insurance and improvement of the developers' ability to meet standards and baselines pertaining to the quality of the software that they develop. Some of the functions of this group include improving the accuracy, knowledge, and performance of individuals and teams, to create the actual standards and baselines required for software quality, and to provide input to external software sources to improve the externally developed software quality, all based upon the collected software quality metrics. If each of the four groups of functions are successfully carried out by their respective parties, this will insure the proper collection, analysis, and utilization of the software quality metrics.

In summary, in today's business model a new breed of software quality metrics are becoming increasingly and drastically needed. The need for new metrics arise from four major problems that stem
from the new business model. The problems are the repercussions due to application malfunction, the necessity for agility in a modern business, the dependence upon software developed by some external source, and the exorbitant costs of maintenance associated with the ownership of an application. To address or avoid these problems one must generate metrics based upon the characteristics of quality software, and utilize those metrics to increase the quality of the software. There are five main characteristics associated with quality software that will account for the problems mentioned above. Those characteristics are transferability, changeability, robustness, performance, and lastly security. For any piece of software, for the most part, each one of these characteristics can be transformed into a metric to be collected and analyzed. Once one derives a new set of metrics based upon these characteristics, it is best to deploy slowly and in small sections of the development process. Once one phase is complete, one can learn from their mistakes and make changes to the deployment process to increase its effectiveness. During the deployment of the new metrics, the responsibilities can be divided up into four groups of functions. The groups are named governance, management, evaluation, and improvement, each of which is dependent upon every other group. Every section of a business is responsible for a different group of functions, and as a result are the sections are dependent upon each other as well. This means that each section must work to collect and utilize these metrics as best they can and also to collaborate with the other sections to improve the software as an entire company. By improving the quality of the software used by the business via the proper collection, analysis, and utilization of software quality metrics based upon software quality characteristics, one can avoid the problems plaguing the modern business model, and provide great benefit to the business. Quality software can provide a business with an insurance of correct functioning, the ability to adapt to changing conditions, the ability to diminish dependence upon external suppliers, and a reduction of the costs of owning an application. All of which will strengthen and better the business.
References

1. Curtis, Bill Dr. “Using application quality metrics to improve the business value of applications”
