MANAGING SOFTWARE QUALITY USING HACKYSTAT

A THESIS PROPOSAL SUBMITTED TO MY THESIS COMMITTEE

MASTERS

IN

INFORMATION AND COMPUTER SCIENCES

By
Aaron A. Kagawa

Thesis Committee:

Philip M. Johnson, Chairperson
...

September 23, 2004
Version 1.1.0

Abstract

This is the abstract of my thesis proposal.

Chapter 1

Introduction

Provide an introduction to whole thesis.

This chapter will introduce you to the main ideas of this thesis. We will first briefly look at what sofware quality is and the problems that is associated with trying to achieve high software quality. Next, I will introduce Hackystat and the Hackystat extension that I will be building to help address some of the problems associated with managing software quality.

1.1 Software Quality has Many Meanings

The term “Software Quality” has many meanings. This is a test citation [2].

Attributes:

1. The degree to which software possesses a desired combination of attributes [3].
2. The composite of all attributes which describe the degree of excellence of the computer system [1].
3. The degree to which a software product possesses a specified set of attributes necessary to fulfill a stated purpose [4].
4. Quality is a collection of attributes: portability, reliability, efficiency, usability, testability, understandibility, and modifiability [**?**]

Others

1. The fitness for use of the total software product [5]

And there is more. People affect quality.

Whatever the definition, developers and managers alike need to strive for the very highest of quality in every project they do. However, if people cannot agree on what quality is then how are they to improve it? When the very definition of something is faulty anything built upon that will very likely be faulty as well. This is the bane of any quality assurance program.

This isn't to say that there have been

1.2 Quality is a Subjective Term

Software Quality has many different meanings because it is subjectively interpreted differently. Like all subjective terms, Software quality means different things to different people. Attempts in the software engineering community to use qunatifiable measures of quality is useful until the exact meaning of quality is challenged by others holding there own subjective meaning. Here lies the extraordinary problem with acheiving high software quality that is recognized by our peers.

In other engineering fields, like Mechanical Engineering, it is much easier to view quality as a qualitative term because the products produced by other fields are physical objects that can be tested. Software on the other hand, is not physical, it can not be picked up and banged against the ground to determine quality. It isn't made up of physical materials like metal. Metal has universally accepted quanitifiable measures and theoritical laws that apply to it. A product that is created with metal thus has the quantifiable measures associated with the material itself. Attempts to fully quanitify software quality will always fail unless it can be constructed with "materials" that have fully quantifiable measures and are universally accepted.

Quality of software is much like evaluating the quality of a cup of coffee. It depends on subjective measures, for example, the aroma of the coffee grinds. We can subjectively sense the differences of a gourmet cup of coffee from those we find in our hotel room. The exact process of that determination is very debatable and is the essence of a subjective measure.

However, subjective measures, like software quality, need not be based soley on our own intuition. As in determining the quality of a coffee grind, we can base our decissions on quantitative data. For example, the price of a coffee grind provides a quantifiable measure on which we base our subjective view. This isn't to say that the price accurately reflects the "quality" of a coffee grind but from our experiences we assume it to be true.

Why attempt to quantify the un-quantifiable? Why try to quanitify the quality of coffee when we already have a good understanding of how good a cup of coffee is using our own subjectivity? Until someone invents a universally acceptable quantifiable measure of a subjective meaing, we should embrace its existance instead of ignoring it. As I previously stated, it isn't the case that quantifiable measures of other aspects of a product is invalid. We should use these measures to form our subjective view of quality.

In this thesis I will attempt to use quantifiable measures of software to determine subjective quality. The important distiction between previous works in this field is that subjectivity will be modeled into the quality measure itself.

1.3 Some Software Quality Measures

This section provides some software quality measures that are in use.

Defect Density

1.4 The Hackystat Quality System: a system to manage the quality of a project

1.5 Thesis Statement

This research investigates the

Bibliography

[1] M. J. Fisher and W. R. Light Jr. *Definitions in Software Quality Management*. Petrocelli Books, New York, 1979.

[2] Watts Humphrey. The IBM large-system software development process: Objectives and direction. *IBM Systems Journal*, 24(2), 1985.

[3] Software Engineering Technical COmmittee of the IEEE Computer Society. IEEE Standard Glossary of Software Egineering Terminology. *IEEE-STD-729-1983 (New York; IEEE)*, 1993.

[4] D. J. Reifer. *State of the Art in Software Quality Management*. Reifer Consultants, Torrance, 1985.

[5] G. G. Schulmeyer and J. I. McMancus. *Handbook of Software Quality Assurance*. Van Nostrand Reinhold, New York, 1987.