Orthogonal Defect Classification—A Concept for In-Process Measurements

Ram Chillarege, *Senior Member, IEEE,* Inderpal S. Bhandari, *Member, IEEE,* Jarir K. Chaar, *Member, IEEE,* Michael J. Halliday, Diane S. Moebus, Bonnie K. Ray, and Man-Yuen Wong

Abstract—**This paper describes orthogonal defect classification (ODC), a concept that enables in-process feedback to developers by extracting signatures on the development process from defects. The ideas are evolved from an earlier finding that demonstrates the use of semantic information from defects to extract cause-effect relationships in the development process. This finding is leveraged to develop a systematic framework for building measurement and analysis methods. This paper**

- **defines ODC and discusses the necessary and sufficient conditions required to provide feedback to a developer;**
- **illustrates the use of the defect type distribution to measure the progress of a product through a process;**
- **illustrates the use of the defect trigger distribution to evaluate the effectiveness and eventually the completeness of verification processes such as inspection or testing;**
- **provides sample results from pilot projects using ODC;**
- **opens the doors to a wide variety of analysis techniques for providing effective and fast feedback based on the concepts of ODC.**

I. Introduction

IN RECENT years the emphasis on software quality has increased due to forces from several sectors of the computer industry. Software is one of the slowest processes in enabling new business opportunities, solutions, or dimensions of computing, and is a significant part of total cost. It has also been found that the dominant cause of system outage has shifted to software given that it has not kept pace, in the past few years, with improvements in hardware or maintenance [1]. Thus there is a resurgence of research in software topics such as reliability, engineering, measurement, etc. [2], [3]. However, the area of software quality measurements and quantification is beset with undue complexity and has, in some ways, advanced away from the developer. In an area where the processes are so amorphous, the tangibles required for measurement and modeling are few. With the result academic pursuits that can't be confined to the limitations of practice evolved and became distanced from the developer. In this area, the need to derive tractable measurements that are reasonable to undertake and intuitively plausible cannot be understated. Measurement without an underlying theme can leave the experimentalist, the theorist and the practitioner very confused.

Manuscript received October 1, 1991; revised August 1, 1992. Recommended by R. Selby and K. Torii.

R. Chillarege, I.S. Bhandari, J.K. Chaar, M.J. Halliday, B.K. Ray, and M.-Y. Wong are with IBM T.J. Watson Research Center, Yorktown Heights, NY 10598.

D.S. Moebus is with IBM Mid-Hudson Valley Programming Lab, Wappinger Falls, NY.

IEEE Log Number 9203764.

The goal of this paper is to develop reasonable measurements. It does this by examining the fundamental properties of such measurements and then deriving the rationale for analysis and models. To put the domain of software in-process measurements and analysis in perspective, let us examine two extremes of the spectrum: statistical defect models and qualitative causal analysis.

1.1. Statistical Defect Models

The goal of statistical defect modeling, which includes what is commonly referred to as software reliability growth, has been to predict the reliability of a software product. Typically, this may be measured in terms of the number of defects remaining in the field, the failure rate of the product, the short term defect detection rate, etc. [3]-[5]. Although this may provide a good report card, it often occurs so late in the development cycle that it is of little value to the developer. Ideally, a developer would like to get feedback during the process.

1.2. Causal Analysis

The goal of causal analysis is to identify the root cause of defects and initiate actions so that the source of defects is eliminated. To do so, defects are analyzed, one at a time, by a team that is knowledgeable in the area. The analysis is qualitative and only limited by the range of human investigative capabilities. To sustain such pursuit and provide a focus for the activity a process description has been found useful. At IBM, the Defect Prevention Process [6] and similar efforts elsewhere have found causal analysis to be very effective in reducing the number of errors committed in a software project. The qualitative analysis provides feedback to developers that eventually improves both the quality and the productivity of the software organization [7].

Defect Prevention can provide feedback to developers at any stage of their software development process. However, the resources required to administer this method are significant, although the rewards have proven to be well worth it. Moreover, given the qualitative nature of the analysis, the method does not lend itself well to measurement and quantitative analysis. Consequently, Defect Prevention, though not a part of the engineering process control model, could eventually work in conjunction with it.