A Framework for Measuring and Evaluating Program Source Code Quality

Hironori Washizaki[1], Rieko Namiki[2], Tomoyuki Fukuoka[2], Yoko Harada[2], and Hiroyuki Watanabe[2]

[1] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
washizaki@nii.ac.jp
[2] Ogis-RI Co., Ltd., MS-Shibaura Bldg., 13-23, Shibaura 4, Minato-ku, Tokyo, Japan
{Namiki_Rieko,fukuoka_tomoyuki,Harada_Yoko,Watanabe}@ogis-ri.co.jp

Abstract. The effect of the quality of program source code on the cost of development and maintenance as well as on final system performance has resulted in a demand for technology that can measure and evaluate the quality with high precision. Many metrics have been proposed for measuring quality, but none have been able to provide a comprehensive evaluation, nor have they been used widely. We propose a practical framework which achieves effective measurement and evaluation of source code quality, solves many of the problems of earlier frameworks, and applies to programs in the C programming language. The framework consists of a comprehensive quality metrics suite, a technique for normalization of measured values, an aggregation tool which allows evaluation in arbitrary module units from the component level up to whole systems, a visualization tool for the evaluation of results, a tool for deriving rating levels, and a set of derived standard rating levels. By applying this framework to a collection of embedded programs experimentally, we verified that the framework can be used effectively to give quantitative evaluations of reliability, maintainability, reusability and portability of source code.

1 Introduction

In today's world, where value is controlled in every corner of society by software systems from the embedded to enterprise level, demand is increasing for a system of technology to measure and evaluate system quality characteristics (e.g. reliability) to use the evaluation results to maintain and improve the system. In this paper we propose a quality evaluation framework based on quantitative quality measures, for software engineers involved in development, maintenance or procurement of software, or others involved in improvement of development processes. We deal with quantitative measures of quality that take measurements of program source code written in the C programming language.

There is great demand for practical technologies which can measure and evaluate quality with high precision and identify quality characteristics that will cause problems or will need improvement, because the quality of the source code has a significant effect on the overall system performance and cost of development and maintenance. In the past, various techniques for measuring quality

J. Münch and P. Abrahamsson (Eds.): PROFES 2007, LNCS 4589, pp. 284–299, 2007.

have been proposed, but they generally have not covered quality characteristics comprehensively and the metrics or measured results have not been widely used[1].

In response, we propose a framework which applies to source code written in the C programming language and implements quality measurements and evaluation effectively. The framework is independent of any person/evaluator properties, and resolves the problems of conventional approaches.

2 Problems with Conventional Quality Measurements

From a quality point of view, measurement methods can be classified into four types based on amount of information. A *Metric* contains the least amount of information, and simply measures a particular property without relating it to quality. A *Quality Metric* measures a property and includes a way to interpret the measurement result in terms of a quality characteristics. *Quality Metrics* is used to refer to multiple such metrics for a single quality characteristics and a *Quality Metrics Suite* treats several quality metrics and systematically summarizes the results of each.

Though many metrics have been proposed, it is generally difficult to select an appropriate one from among them or to interpret the measurement results[2]. Further, they and measured values have not been broadly useful[1]. For quality measures which apply to source code in particular, the main problems are summarized below.

(P_1) **Non-comprehensive suites:** In order to take into account tradeoffs between different quality characteristics (e.g. time-behaviour vs. analysability), it is desirable to be able to measure and evaluate all quality characteristics, which effect the final system's quality in use, at the same time. However, most of the existing metrics which apply to source code do not relate measurement values to quality, or provide a quality metric which measures quality based on only a single characteristic. There are a few suites which handle source code, including REBOOT[3], QMOOD[4], SPC suite[5], the suite from Ortega[6], the EASE project result[7] and the ISO/IEC TR 9126-3 reference implementation[8]; however, they all require additional input besides the source code (e.g. a design model) and/or they lack comprehensive coverage of the measurable and assessable source code characteristics specified by the ISO9126-1[9] (or equivalent) quality model.

(P_2) **Lacking in ability to break-down or overall evaluation:** Generally, source code written in a high-level programming language has a layered structure, with inclusion relationships between multiple logical and physical modules. For example, in the C programming language, generally functions are included in files, files in directories, and directories in the system, giving a four-layer structure. In this case, it is desirable to measure and evaluate the quality of individual module units according to various objectives, such as comparing the entire integrated system quality with another system and evaluating the quality