Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information & Management 40 (2003) 799–812
INFORMATION & MANAGEMENT
www.elsevier.com/locate/dsw

Quality management metrics for software development

John S. Osmundson[a,1], James B. Michael[b,*], Martin J. Machniak[c,2], Mary A. Grossman[d,3]

[a]Department of Information Sciences, Naval Postgraduate School, CC/Os, 833 Dyer Road, Monterey, CA 93943, USA
[b]Department of Computer Science, Naval Postgraduate School, CS/Mj, 833 Dyer Road, Monterey, CA 93943, USA
[c]Space and Naval Warfare Systems Center, OTC-2, Code 2334, 53560 Hull Street, San Diego, CA 92152, USA
[d]NASA Dryden Flight Research Center, P.O. Box 273, MS 4840A, Edwards, CA 93523, USA

Received 22 July 2001; received in revised form 15 April 2002; accepted 23 August 2002

Abstract

It can be argued that the quality of software management has an effect on the degree of success or failure of a software development program. We have developed a metric for measuring the quality of software management along four dimensions: requirements management, estimation/planning management, people management, and risk management. The quality management metric (QMM) for a software development program manager is a composite score obtained using a questionnaire administered to both the program manager and a sample of his or her peers. The QMM is intended to both characterize the quality of software management and serve as a template for improving software management performance. We administered the questionnaire to measure the performance of managers responsible for large software development programs within the US Department of Defense (DOD). Informal verification and validation of the metric compared the QMM score to an overall program-success score for the entire program; this resulted in a positive correlation.

Keywords: Metrics; Software management; Software process

1. Introduction

Quality of management has long been a concern within the software engineering community. The term ''software crisis'' was coined in the 1960s to refer to problems in developing software on time, within budget, and with the properties that the software was usable and actually used. The General Accounting Office reported in 1979 [23] that of the government software development projects studied:

- more than 50% had cost overruns;
- more than 60% had schedule overruns;
- more than 45% of the delivered software could not be used;
- more than 29% of the software contracted for was never delivered;
- more than 19% of the delivered software had to be reworked.

* Corresponding author. Tel.: +1-831-656-2655; fax: +1-831-656-2814.
E-mail addresses: josmundson@nps.navy.mil (J.S. Osmundson), bmichael@nps.navy.mil (J.B. Michael), machniak@spawar.navy.mil (M.J. Machniak), mary.grossman@mail.dfrc.nasa.gov (M.A. Grossman).
[1] Tel.: +1-831-656-3775; fax: +1-831-656-3679.
[2] Tel.: +1-619-524-3473; fax: +1-619-524-3507.
[3] Tel.: +1-661-276-5531; fax: +1-661-276-2792.

0378-7206/$ – see front matter
PII: S0378-7206(02)00114-3

Since that report was released, the software engineering community has attempted to improve the software development process. For example, the well-known capability maturity model (CMM) identifies key practices required to improve organizations' software development processes. The CMM was codified into five levels of increasing organizational maturity. More recently a personal software process (PSP) has been introduced, defining key practices required to improve an individual's software development processes [11]. There is evidence that attaining high levels of capability as defined by the CMM has resulted in improvements in the management of the development of software-based systems [3] and further evidence has shown the potential benefits to be derived by enacting a PSP [7].

The quality of software development management tools has improved over the past 30 years. However, many of the same challenges, such as keeping software development projects on schedule and within budget, remain today. The Standish Group report [9] in 1995 found that, on average, approximately 16% of software projects were completed on time and within budget. In large companies the record was even worse: only 9% of the projects were completed on time and within budget. Moreover, the projects that were completed contained only approximately 42% of the originally proposed features and functions. Also, US-based companies and government agencies spent $ 81 billion for canceled software projects and these same organizations paid an additional $ 59 billion for software projects that were completed, but exceeded their original time estimates. According to Fabian-Isaacs and Robinson [5], ''software development projects are notorious for running over budget and behind schedule''. The Center for Project Management in San Ramon, CA reported that 99% of commercial software products are not completed on time, within budget, or according to specifications, and that the average project is underestimated by 285%.

CMM level-four compliance requires the development organization to collect metrics that measure the effectiveness of the development process, while CMM level five requires that the organization use the metrics continuously to improve its development process. IEEE Standard 12207.0 and the earlier MIL-STD-498 include metrics that might be gathered during the software development process. Examples include software size and complexity, software units developed over time, milestone performance, and problem/change report status. Metrics are defined for the software development process and the software product but not for the quality of the project management. One can argue that in order to systematically go about improving the management of software projects, it is necessary to measure the quality of project management. Program-management tools have been developed to assist the program manager in estimating the cost and schedule of software programs. However, the estimation tools available assume consistent and high-quality program management.

One of the earliest and most widely used software project cost-estimation models is COCOMO [2]. The basic, intermediate, and detailed COCOMO models are based on the results of analyzing 63 software projects and applying regression analysis in order to predict software development cost as a function of software size and other factors. The intermediate and detailed COCOMO models take into account attributes of the software product, computer hardware, development personnel, and the project. Examples of project attributes include the use of software tools and the required development schedule. Intermediate cost estimates are based on the estimated number of lines of code (LOC) to be developed and then these estimates are adjusted by applying multipliers determined by rating the project with respect to the attributes. For example, a project completed under an accelerated schedule is estimated to cost more. However, COCOMO does not take into account the quality of project management.

Boehm wrote, ''poor management can increase software costs more rapidly than any other factor'' and ''despite this cost variation COCOMO does not include a factor for management's quality, but instead provides estimates which assume that the project will be well managed''. Among the reasons Boehm gave for not including management quality was that management quality ratings are not easy to determine.

If the quality of the software program management were measurable and available as input to costing and scheduling tools, the resulting estimates could pinpoint areas of software program management in which improvement needs to be made. Being able to measure the quality of management of software projects objectively allows development of more accurate cost models and would also provide a means for improving