Does XP Deliver Quality and Maintainable Code?

Raimund Moser, Marco Scotto, Alberto Sillitti, and Giancarlo Succi

Center for Applied Software Engineering, Free University of Bolzano-Bozen,
Piazza Domenicani 3, Italy
{rmoser,mscotto,asillitti,gsucci}@unibz.it

Abstract. Extreme Programming aims at delivering working software for less money and still of high quality. It is well known that software maintainability is one of the most important concerns and cost factors of the software industry. The question of this research is whether Extreme Programming intrinsically delivers easily maintainable code or not. We propose a model on how to evaluate the evolution of source code quality and in particular maintainability in an Extreme Programming environment and evaluate it with a small case study. The results obtained from the case study seem to sustain the hypothesis that Extreme Programming enhances quality and in particular maintainability of a software product. Given such promising results, additional experimentation is required to validate and generalize the results of this work.

Keywords: quality, maintainability, metrics.

1 Introduction

In Extreme Programming (XP) much emphasis is given on an agile, iterative and customer oriented way of how to develop software. Among the top priorities of XP are to satisfy the customer through continuous delivery of valuable software and to welcome changing requirements (http://agilemanifesto.org). XP practices are tailored to achieve such goals: iterative and informal planning, simple design, continuous refactoring of the code, pair programming, test first and continuous integration – just to mention a few [2]. Most of these practices are intended to be used during development and maintenance and seem to keep at least in part their promises [13], [18]. Although Kent Beck states “Maintenance is really the normal state of an XP project” [2] he is aware of the differences and problems of a system under development and a delivered system. He suggests that the effort for changing code in production is almost twice of ideal engineering time and that also XP cannot avoid the entropic death of a software system – all it can do is to extend its lifetime as far as possible.

Therefore, we think that also for XP projects maintainability is a key concern and quality factor. Fred Brooks already claimed [4]: “The total cost of maintaining a widely used program is typically 40 percent or more of the cost of developing it”. Other researchers confirmed recently such numbers [7]. Therefore, high maintainability is a long-term success factor for a software product. We think that XP intrinsically guides software engineers to develop products, which are likely to show good quality and maintainability.
Maintainability is a high-level quality metric that combines several internal and external properties of a software product and of the development process [9]. To assess maintainability in this research we use only internal product attributes that are available during development and we monitor their evolution over time. We do not take into account any external product or process metrics.

The paper is organized as follows. In Section 2, we present our research methodology and propose a model for a Maintainability Trend indicator. In Section 3 a case study is presented and discussed; in Section 4 we touch on some issues regarding the limitations of our approach and our future plans. Finally, conclusions and implications of the investigation are drawn in Section 5.

2 A Model to Assess the Evolution of Source Code Quality and Maintainability

In this section we describe the metrics used for assessing maintainability. Afterwards, we develop a model for evaluating how the maintainability of a software system evolves during development.

2.1 Internal Product Metrics That Affect Quality and Maintainability

Our research question is to assess whether XP facilitates the development of high maintainable code or not. Maintainability is a rather vague term for describing certain quality attributes of a software system and can be decomposed into lower-level metrics in different ways [9]. In this research we focus only on internal properties of the software that are considered to be relevant for its maintainability. We use three major sources to identify the metrics to use:

- Metrics used to assess testability of object-oriented software [15], [5]
- The Maintainability Index (MI) proposed by Oman [17]

The motivation for choosing this set of metrics is twofold. First, some of them such as the CK metrics are among the best-understood and validated metrics for object-oriented systems, therefore we can be more confident in their expressiveness. Second, the tool we use for collecting these metrics is able to collect them in an automatic and in a non-invasive way - a fundamental requirement for data collection in an XP process [12].

Several empirical studies put the CK metrics into relationship with software quality and maintenance. Li and Henry [14] for example show that the CK metrics are useful to predict maintainability. Basili et al. [3] investigate the relationship between the CK metrics and code quality. Their findings suggest that 5 of the 6 CK metrics are useful quality indicators. However, such studies are rare in XP-like environments and they do not analyze the evolution of the CK metrics during development to see whether there is an observable trend towards an enhancement or degradation of the software in terms of these metrics.

Testability is an important sub aspect of maintainability. Different models have been proposed on how to measure testability in object-oriented systems. Checking the