Information Sciences 328 (2016) 237–249

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Assessment of software developed by a third-party: A case study and comparison

Tadas Remencius [a], Alberto Sillitti [b], Giancarlo Succi [c,*]

[a] Connexx S.r.L., Bolzano/Bozen, Italy
[b] Free University of Bolzano/Bozen, Italy
[c] Innopolis University, Russian Federation

ARTICLE INFO

Article history:
Received 9 December 2014
Revised 18 July 2015
Accepted 9 August 2015
Available online 24 August 2015

Keywords:
Empirical software engineering
Software metrics

ABSTRACT

Most of the research effort in the area of software analysis is focused on the perspective of the developer (as in "software developing company") and the ways how the software development process could be improved. However, that is not the only type of software assessment common in the industry. There are also assessments that are commissioned by other parties, such as the primary recipients of the software solutions or courts dealing with legal cases that are related to software products or services. This work presents one such case-study that was performed for a public administration in Italy. The paper describes the assessment itself and also points out the need for more focused research by providing a comparison between developer-oriented and customer-oriented assessment types.

1. Introduction

Improvement of software and software development process requires sufficient level of understanding of what exactly is happening and why. As such, analysis of produced software and of the processes related to its development play a critical role both in the scientific area of software engineering and in practice. Afterall, it is very difficult, if not impossible, to control what cannot be understood. Unsurprisingly, a lot of attention is being spent by researchers on different low level aspects of software (e.g., code metrics) and on development process itself (e.g., process models and process metrics). Unfortunately, the amount of reported case-studies and experiences from the industry is relatively small. Furthermore, such studies primarily focus on a scenario where the main recipient of the analysis is the developer of the software. However, in practice there is also a different scenario that is quite common: when the assessment of the software is commissioned by non-developer. In such cases, the typical recipient of the analysis is either the main customer of the software system considered or a third-party, such as the court handling a legal case. Such assessments are quite different from those where the client is the developer. While the core of the analysis is still performed on the same target software, the focus and the objectives of such assessments are what set them apart. In order to better highlight and understand these differences, we propose to look at such cases as a separate type of software assessments. Within the scope of this paper, we classify software assessments into two categories:

1. "Developer-oriented" – assessments ordered by the developer of the software.
2. "Customer-oriented" – assessments ordered by a third-party.

* Corresponding author at: Innopolis University, Software Engineering Institute, Universitetskaya Streeet 1, 420500 Innopolis, Republic of Tatarstan, Russian Federation. Tel.: +393245312230.
E-mail addresses: Tadas.Remencius@gmail.com (T. Remencius), asillitti@gmail.com (A. Sillitti), Giancarlo.Succi@acm.org (G. Succi).

http://dx.doi.org/10.1016/j.ins.2015.08.028
0020-0255/

This paper describes one such customer-oriented assessment and also provides a comparison between the two assessment types. The goal is to contribute to the body of knowledge on this topic by presenting a case-study from a public administration, and to encourage discussion and research focused more on the conceptual aspects of software assessments.

The industrial software assessment presented in this paper took place in Italy. It was commissioned by one of the public administrations. The target of the assessment was a complex supplier and data management system that was developed by an independent software development company that had won the public competition.

The provided comparison of assessment types is based on the personal experience of the authors gained from multiple previous software and software development process assessment projects with industrial partners.

The paper is structured as follows: In Section 2, covers important related works by other researchers as well as previous publications on the topic written by the authors; Section 3 describes the details of the assessment; Section 4 provides the general comparison between developer-oriented and customer-oriented type of assessments; Section 5 concludes with a brief summary of the paper.

2. Related work

As far as we are aware, there are no scientific publications about the assessments of software when such assessments are ordered by non-developer. However, a number of papers highlight certain aspects that we feel are important to such type of analysis or illustrative of its specific characteristics.

2.1. Works by others researchers

A large amount of software engineering research is focused on software metrics [17]. A number of papers have been published on this topic over the years, but the main question is still unanswered: how exactly can we use software metrics in practice to get real value for the business? A wide range of metrics has been identified and there are Open Source tools available on the market to compute most of them [15]. Unfortunately, it is still unclear how to reliably interpret specific metric values for particular software systems to achieve visible benefits.

Benchmarking is one of the approaches that could provide answers or at least give useful hints for this. One of the more recent works in this direction is that of Lochmann [16]. The author explored a benchmarking-based approach to determine threshold values for metrics. Based on his findings the author concluded that benchmarking is a feasible approach for threshold identification, where the size of the benchmarking base is the critical factor: “the bigger the base, the smaller is the influence of the randomly chosen systems”. Furthermore, “for a randomly generated benchmarking base of sufficient size, neither the actual selected system, nor the size of the systems contained in it, has a major influence”.

The work of Ferreira et al. [13] is an example of an attempt to identify useful thresholds for specific software metrics. The authors analyzed a large sample (40) of Open Source programs written in Java to determine potential thresholds for six object-oriented software metrics. Such thresholds should indicate the good and/or bad ranges of metric values, thereby assisting in automatic identification of problematic code components (e.g., classes).

Another important research direction is in how to combine or aggregate metrics to assess higher level characteristics of software or software processes. An example of such work is that of Mordal et al. [18]. It contributes towards making low-level software metrics (e.g., defined at the level of individual components, such as classes) useful for the assessment of the quality attributes of the overall software systems by deriving requirements for a metric aggregation method. The authors also propose their own aggregation model, called Scale [18].

Software quality models take a different, top-down approach by building a scheme of how software quality can be broken down into various characteristics and relationships among them [28,29]. The paper of Dubey et al. [12] provides a comparison of such models.

What the quality models are lacking, however, is the link to the actual metrics. The work such as those of Maurer et al. [17] and Dubey and Rana [10,11] attempt to fill that gap.

However, understanding the general meaning of metric values is just the first step. It is just as (if not more) important to be able to analyze the implications of those meanings and to identify effective steps to make use of those findings. As of now, this is where the human experts are mandatory.

The critical role of experts is illustrated by the work of Dubey et al. [12], which describes a goal-oriented expert-based software design assessment method called MIDAS (Method for Intensive Design Assessments) [23] that was developed within Siemens Corporate Development Center Asia Australia to address the need of more efficient design assessment approach. Application of MIDAS to three different projects showed that the third-party experts played a critical role in making the design assessments effective. The experts were the key factor that allowed to select and use the proper software analysis tools efficiently, by ensuring constant focus on the assessment objectives and the relevant design constraints to extract the information that was really important for the task (e.g., filtering-out the false positives, selecting the indicative metrics, etc.).

The downside of expert-based assessments is that they imply manual involvement and, as such, tend to be much more time consuming and costly. A lot depends on the expertise and professionalism level of the expert. The work of Sauer et al. [14,25], for example, identified that “the most important factor in determining the effectiveness of SDTRs” (Software Development Technical Reviews) “is the level of expertise of the individual reviewers”.