Information and Software Technology 56 (2014) 477–494

Contents lists available at ScienceDirect

Information and Software Technology

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

Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process

Ayca Tarhan *, Seda Gunes Yilmaz [1]

Hacettepe University Computer Engineering Department, Beytepe Kampusu, 06532 Ankara, Turkey

ARTICLE INFO

Article history:
Available online 16 December 2013

Keywords:
Empirical method
Quantitative analysis
Qualitative analysis
Software measurement
Process performance
Agile development

ABSTRACT

Context: Although Agile software development models have been widely used as a base for the software project life-cycle since 1990s, the number of studies that follow a sound empirical method and quantitatively reveal the effect of using these models over Traditional models is scarce.
Objective: This article explains the empirical method of and the results from systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process adapted in two projects of a middle-size, telecommunication software development company. The Incremental Process is an adaption of the Waterfall Model whereas the newly introduced Agile Process is a combination of the Unified Software Development Process, Extreme Programming, and Scrum.
Method: The method followed to perform the analyses and comparison is benefited from the combined use of qualitative and quantitative methods. It utilizes; GQM Approach to set measurement objectives, CMMI as the reference model to map the activities of the software development processes, and a predefined assessment approach to verify consistency of process executions and evaluate measure characteristics prior to quantitative analysis.
Results: The results of the comparison showed that the Agile Process had performed better than the Incremental Process in terms of productivity (79%), defect density (57%), defect resolution effort ratio (26%), Test Execution V&V Effectiveness (21%), and effort prediction capability (4%). These results indicate that development performance and product quality achieved by following the Agile Process was superior to those achieved by following the Incremental Process in the projects compared.
Conclusion: The acts of measurement, analysis, and comparison enabled comprehensive review of the two development processes, and resulted in understanding their strengths and weaknesses. The comparison results constituted objective evidence for organization-wide deployment of the Agile Process in the company.

1. Introduction

Software development activities such as requirements analysis, design, implementation, testing, and maintenance are carried out within a software development life-cycle in an organization [1]. The life-cycles are assumed to be adapted from software development models that have been implemented for years and proved successful. However, many software projects adapting different development models for their life-cycles fail to achieve their targets. In accordance to an analysis performed by The Standish Group in 2009 [2], only 32% of software projects were reported as successful in comparison to reported 44% as challenged (late, over budget and/or with less than the required features and functions) and 24% as failed (cancelled prior to completion or delivered and never used).

Low success rates of software projects motivated inquiry of Traditional (Plan-driven) software development models in 1990s and in the following years, Agile software development models were proposed as an alternative. The Traditional models were defined to value fully specified problems, rigorous planning, pre-defined processes, and regular documentation [1]. The Agile models, on the other hand, were defined to value individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [3]. Although Agile software development models have been widely used as a base for the software project life-cycle since 1990s, the number of studies that quantitatively reveal the effect of using these models over Traditional models on development performance is scarce. In their systematic review on empirical studies of Agile software development, Dybå and Dingsøyr [4] claim the immaturity of the methods,

* Corresponding author. Tel.: +90 312 2977500; fax: +90 312 2977502.
E-mail addresses: atarhan@cs.hacettepe.edu.tr (A. Tarhan), sedagunes@gmail.com (S.G. Yilmaz).
[1] Tel.: +90 312 2977500; fax: +90 312 2977502.

0950-5849/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2013.12.002

the data, and the results in the limited number of reported studies. Petersen and Wohlin [5] emphasize the need to investigate Agile and incremental models using sound empirical methods, and report a case study revealing the effect of moving from a Plan-driven to an Incremental and Agile development approach. Van Waardenburg and van Vliet [6], on the other hand, investigate the challenges that are brought by the co-existence of Agile models and Plan-driven development and rationalize how two organizations deal with those challenges.

Quantitatively revealing the effect of using software development models on development performance and product quality in various contexts is among the major needs of the software community. However, a systematic review on measurement in software engineering by Gómez et al. [7] claims that software process is the least measured entity (with 21%) in the reported studies. This is mostly due to the diverse issues to consider while empirically measuring and analyzing the performance of a software development process. In addition to having knowledge on process management, measurement, and statistics, this work requires a series of tasks to be carried out such as; identifying the purpose of the analysis, capturing process context, identifying process components, ensuring consistency of process executions, gathering process data, selecting process measures, determining analysis methods, conducting the analysis, and interpreting analysis results. When comparing the performances of two or more development processes, reconciling the scopes of the processes as a base for the comparison is another task to accomplish. Aside from these challenges, there are only a few systematic and practical methods [8,9] that can be utilized as a guide while measuring the performance of a software development process.

In this paper, we explain the steps of and the results from systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process that are both adapted in a middle-size, telecommunication software development company employing 65 developers. The Incremental Process which is in use for 16 years is an adaption of the Waterfall Model whereas the newly introduced Agile Process is a combination of the Unified Software Development Process, Extreme Programming, and Scrum. The projects in which the Incremental and Agile processes were employed have been carried out to create a product family, and the development of 20 features in each project were taken as the base.

The method followed to perform the analyses and comparison proposes the utilization of qualitative and quantitative methods together to derive the conclusions. This paper describes the empirical method in detail and presents example outputs from its application, which is hardly met in the literature. The method utilizes; Goal Question Metric (GQM) Approach [8] to set performance and quality measurement objectives, Capability Maturity Model Integration (CMMI) for Development [10] as the reference model to map the activities of the software development processes, and a pre-defined assessment approach [11] to verify consistency of process executions and evaluate measure characteristics prior to quantitative analysis. The GQM Approach is taken as a guideline to identify measurement goals and required measures, and complemented by a bottom up approach in order to reconcile required measures with available process measures.

The article is organized in seven sections. Section 2 provides background information on the Traditional and Agile software development models, software measurement basics and the GQM Approach, the Capability Maturity Model Integration, the predefined assessment approach called an Assessment Approach for Quantitative Process Management, and the related work on comparative studies of software development models. Section 3 provides an outline of the overall case study including context and scope, research objective and design, and analyses goals and measures. Section 4 describes the analysis method and demonstrates example outputs from the analysis of the Incremental Process. Section 5 presents the quantitative results of the comparison, and discusses the rationale behind these results based on the qualitative data elicited from process enactments of the Incremental and Agile processes. Section 6 elaborates threats to validity for the case study and Section 7 provides the conclusions.

2. Background

2.1. Software development models

The organization of the processes and activities that are used to develop a software product is known as "software development life cycle (SDLC)". The SDLC is aimed to ensure systematic and purpose-oriented development of software products. There are several models that describe how to build this cycle under Traditional and Agile approaches.

In the Traditional approach, process activities are planned in advance and progress is measured against this plan [1]. In the Agile approach, on the other hand, planning is incremental and it is easier to change the process to reflect changing customer requirements. Agile models are aimed to respond to changing requirements, to swiftly produce software, and to make these productions available for the customers [12]. Table 1 provides a summary of the characteristics of the Traditional and Agile models.

2.2. Software measurement and the Goal Question Metric (GQM) Approach

Measurement is vital to understanding, controlling, and improving in an engineering discipline. In software engineering, however, measurement is considered a luxury many times [23]. Because of the abstract and changing nature of the software and the pressure in the timely completion of the software product, management in software engineering is mostly based on observation and assumption, rather than on objective evidence. Creating objective evidence, on the other hand, is not easy and is enabled by adequate resources and training, and by appropriate use of methods and tools. This subsection, therefore, provides an overview of measurement, software measures, and the recommended methods.

Measurement is the process by which numbers and symbols are assigned to attributes of entities in the real world, as to describe them according to clearly defined rules [23]. A measure must specify the domain and the range as well as the rule for performing the measurement mapping. Both entity and attribute to measure should be explicit. Measures can be direct or indirect [23]. Direct measures involve no other attribute or entity, and form the building blocks for assessment. Examples are size, duration, and number of defects. Indirect measures are derived from other measures. Examples include productivity, defect density, and efficiency.

Measurement mapping together with the empirical and numerical relation systems represent the measurement scale [24]. Scales help us to understand which analyses are appropriate on measurement data. Types of scales are nominal, ordinal, interval, ratio, and absolute, in the increasing order of information provided [24]. Nominal scale indicates a difference, just classification, and no ordering (e.g., names of programming languages). Ordinal scale indicates the direction of the difference, and ranking with respect to ordering criteria (e.g., priority assignments of defects). Interval scale indicates the amount of the difference; thus, differences of values are meaningful (e.g., calendar date). Ratio scale indicates an absolute zero; therefore, ratios between values are meaningful (e.g., effort). Absolute scale indicates the number of values (e.g.,