Available online at www.sciencedirect.com
ScienceDirect
Information and Software Technology 50 (2008) 1088–1097
INFORMATION AND SOFTWARE TECHNOLOGY
www.elsevier.com/locate/infsof

Towards software process patterns: An empirical analysis of the behavior of student teams

Éric Germain, Pierre N. Robillard *

Department of Computer and Software Engineering, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal, Que., Canada H3C 3A7

Received 2 May 2006; received in revised form 19 October 2007; accepted 23 October 2007
Available online 7 November 2007

Abstract

Traditional software engineering processes are composed of practices defined by roles, activities and artifacts. Software developers have their own understanding of practices and their own ways of implementing them, which could result in variations in software development practices. This paper presents an empirical study based on six teams of five students each, involving three different projects. Their process practices are monitored by time slips based on the effort expended on various process-related activities. This study introduces a new 3-pole graphical representation to represent the process patterns of effort expended on the various discipline activities. The purpose of this study is to quantify activity patterns in the actual process, which in turn demonstrates the variability of process performance. This empirical study provides three examples of patterns based on three empirical axes (engineering, coding and V&V). The idea behind this research is to make developers aware that there is wide variability in the actual process, and that process assessments might be weakly related to actual process activities. This study suggests that in-process monitoring is required to control the process activities. In-process monitoring is likely to provide causal information between the actual process activities and the quality of the implemented components.

Keywords: Software engineering process; Process patterns; Process activities; Process monitoring; Effort; Empirical study; Software engineering education; Project control and modeling; Process measurement

1. Introduction

Implicit and explicit software engineering processes are an integral part of any software development project. Software processes may range from ad hoc personal practices to an organization-wide process structure with formal maturity assessment. There is a general consensus that explicit software processes are likely to improve the efficiency of software development activities. Their ultimate benefit should be a better understanding of software development processes, resulting in improved software quality products and more realistic estimations of the budgeted effort.

Process assessments are based on a Process Reference Model (PRM). A PRM provides descriptions of the process entities to be evaluated and defines what is to be measured. A major asset of the PRM is that it provides a common terminology and scope description for process assessment. A Process Assessment Model (PAM) supports the conduct of an assessment [1].

A basic premise of the PAM is that the quantitative score from the assessment is associated with the performance of the organization or project. The predictive validity of the process capability score is that improving the software engineering practices according to the assessment model is expected to subsequently improve the performance of the product quality. The rating unit in the PAM is the process instance, which is defined as a singular instantiation of a process that is uniquely identifiable and about which information can be gathered in a repeatable manner [2].

This study is not concerned with the assessment of the process instance, but rather by the use of prescribed

* Corresponding author. Tel.: +1 514 340 4238; fax: +1 514 340 3240.
E-mail address: Pierre-n.robillard@polymtl.ca (P.N. Robillard).

0950-5849/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2007.10.018

process instances by various developers. The idea is to illustrate that, although the PAM has assessed the process instance at a specific level, there may be great variability in the use or implementation of various process instances.

Fig. 1 illustrates that a process is composed of various process instances, which can be assessed through a PAM with regard to some PRM, the two most popular PRMs being ISO/IEC 15504 and CMMI. The PAM is often used to validate a predictive model based on some quality metrics of the product or performance indicators of the project.

Our work can be positioned with respect to the process-centered environment in the following way [3]. Processes are defined according to three levels of abstraction. The process model definition contains characterizations of processes expressed in a process modeling language. Process model enactment encompasses the manual or automatic execution of an instantiated model, and defines a variety of means to support software process performers. Process performance encompasses the actual tasks and activities that are performed by the human, who may play various roles. Process enactment and process performance have to be coupled, in order to provide relevant and useful support for software development. In the real world, process enactment and process performance deviate, which leads to a definition of the actual process as encompassing the activities that are really performed, and of the observed process as depending entirely on the performer's feedback [4]. This empirical study is aimed at measuring the actual process based on the performer's feedback.

The success of a software process implementation depends mostly on human understanding, acceptance and realization of the various activities defined by the process. We realize, by observing various software development organizations, that sometimes the real activities performed by the team-mates are quite different from the prescribed process activities.

This paper presents observatory studies in which process activities are recorded while projects are running. Analyses

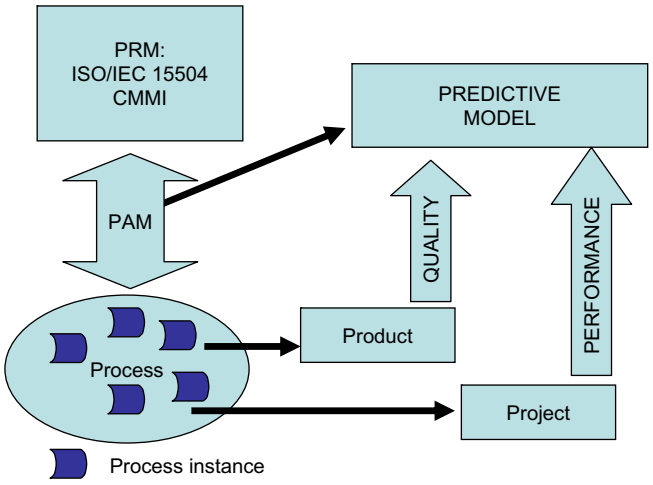

Fig. 1. Predicitive model based on process assessment.

of these data elements provide some insight into the variability of the process activities as performed by different teams instructed to use the same prescribed process. The purpose of this paper is to illustrate through these studies the discrepancies that might occur between the reality of software development and the software process definitions. It stresses the need for measurements to validate that the activities are really performed as prescribed, and presents a new graphical representation that shows the patterns of process activities as the project progresses.

Software processes could be monitored and controlled in just the same way as any manufacturing or business process. Florac and Carleton [5] state that software processes can perform poorly, or they can be unstable, non-compliant or incapable of delivering products that meet requirements. Unstable processes can sometimes be stabilized by identifying the possible causes of the instability and taking steps to eliminate those causes. Processes lacking in capability will require modification and then stabilization, since modification alone may destabilize a process (chapter 7 of [5]). The possibilities offered by process monitoring are discussed extensively in [6,7].

In a manufacturing environment, outlier data points on a control chart representing some quality evolution characteristics may provide indications of anomalies in process execution. Thus, one can look for cases where data points are outside the control ranges or expected evolution pattern, and then takes appropriate action. In manufacturing environments, control charts often monitor the quality characteristics of artifacts, such as part dimension. Under expected conditions, those quality characteristics will exhibit similar values. In the field of software engineering (or in any other branch of engineering where process activities are important), effort data from various process activities could be appropriate for monitoring process conformance and project evolution.

The use of total effort data from a given project is of little value in process control, since the project must be completed in order for any significant analysis to be performed, and this defeats our purpose. So, we may want to use intermediate effort data, i.e. current effort to date at any point in time. In theory, the actual effort at a given point in time could be compared with estimations, and the difference between the two values measured. This would require the use of a fixed reference point that would enable comparison, such as calendar time or a milestone. However, practical ways of performing such a comparison have not yet been found. For instance, MacDonell and Shepperd [8] have studied the effort expended in sixteen software development projects undertaken by a single organization in order to determine the viability of prior-phase effort analysis for re-estimation in the course of a project. They found "little support for the idea of standard proportions of effort distributed between phases." Therefore, in a phase-based approach, where effort is expended during successive phases of a project, any comparison between projects would likely be meaningless, save for special cases which