Information and Software Technology 59 (2015) 205–221

Contents lists available at ScienceDirect

Information and Software Technology

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

A framework for software process deployment and evaluation

Iván Ruiz-Rube [a,*], Juan Manuel Dodero [a], Ricardo Colomo-Palacios [b]

[a] Superior School of Engineering, Avenida de la Universidad, 10, 11519 Puerto Real, Cádiz, Spain
[b] Østfold University College, B R A Veien 4, 1783 Halden, Norway

ARTICLE INFO

Article history:
Received 23 July 2014
Received in revised form 28 November 2014
Accepted 2 December 2014
Available online 15 December 2014

Keywords:
Software quality
Software Process Engineering
Model-driven engineering
Information integration
Linked open data

ABSTRACT

Context: Software Process Engineering promotes the systematic production of software by following a set of well-defined technical and management processes. A comprehensive management of these processes involves the accomplishment of a number of activities such as model design, verification, validation, deployment and evaluation. However, the deployment and evaluation activities need more research efforts in order to achieve greater automation.
Objective: With the aim of minimizing the required time to adapt the tools at the beginning of each new project and reducing the complexity of the construction of mechanisms for automated evaluation, the Software Process Deployment & Evaluation Framework (SPDEF) has been elaborated and is described in this paper.
Method: The proposed framework is based on the application of well-known techniques in Software Engineering, such as Model Driven Engineering and Information Integration through Linked Open Data. It comprises a systematic method for the deployment and evaluation, a number of models and relationships between models, and some software tools.
Results: Automated deployment of the OpenUP methodology is tested through the application of the SPDEF framework and support tools to enable the automated quality assessment of software development or maintenance projects.
Conclusions: Making use of the method and the software components developed in the context of the proposed framework, the alignment between the definition of the processes and the supporting tools is improved, while the existing complexity is reduced when it comes to automating the quality evaluation of software processes.

1. Introduction

Software Process Engineering is the area of software engineering that promotes the systematic production of software by following a series of well-defined technical and management processes, in order to maximize their quality [1]. Thus, organizations need to have methods, techniques and tools to implement a comprehensive strategy of continual quality improvement of their software development and maintenance processes. A comprehensive software processes management requires tools for designing, verifying, validating, deploying and evaluating processes. However, the supporting tools often do not provide mechanisms to include explicit definitions of the processes, which causes a significant lack of consistency between process models and the actual deployment of the tools.

Moreover, experimentation in Software Engineering is a relatively new discipline, which aims to find quantitative answers to specific questions [2]. In order to pose formal experiments, evidence about the procedures, tools and resources used to perform the software life cycle activities is needed. From the data and the evidence generated in different software tools for supporting the software process, the strengths and weaknesses of the elements involved in the software life cycle can be analyzed. However, designing and constructing automatic evaluation methods of such elements is a highly complex task.

This paper presents a framework for the deployment and evaluation of software processes. Software process deployment refers to the implementation of the definitions of the processes on the operational environment. The deployment of processes covers from organizational aspects, such as the implementation of procedures and the acceptance of commitments; technological aspects, such as the configuration and adaptation of tools; and social aspects, such as teamwork and training, among others [3]. In this paper, we will focus on producing structures for maintaining

* Corresponding author. Tel.: +34 956 483 485; fax: +34 956 015 139.
E-mail addresses: ivan.ruiz@uca.es (I. Ruiz-Rube), juanma.dodero@uca.es (J.M. Dodero), ricardo.colomo-palacios@hiof.no (R. Colomo-Palacios).

http://dx.doi.org/10.1016/j.infsof.2014.12.001
0950-5849/

software documentation and models on supporting tools, such as UML modeling and wiki tools, as well as, for planning software projects on issue-tracking tools.

The evaluation of software processes refers to the set of activities needed for measuring the quality of the processes and their suitability to the execution environment after be deployed. Evaluation is also a broad issue because it involves the evaluation of the activities developed during the execution of the projects, people, tools, etc. This framework enables the conduction of technical reviews on software projects for automatically checking the adherence of the projects to the process descriptions, standards, and procedures defined.

The foundations of this framework lie in the following research hypothesis: (*RH1*) inconsistencies between process definitions and execution of the projects could be minimized, in part, by customizing and adapting the support tools and by creating specific templates for them; and (*RH2*) achieving a global and complete view of the information managed by the support tools would enable automating the quality evaluation in software processes.

This framework is based on the application of two well-known approaches in Software Engineering. First, Model-Driven Engineering (MDE) techniques for automating the deployment of process definitions into support tools, and second, Linked Open Data (LOD) for developing information integration solutions intended for evaluating software processes, using the data managed in those tools. Both techniques have proved to provide significant benefits to the community of software engineering. MDE focuses on the design and transformation of models for improving productivity in software development, whereas LOD simplifies the design of the data integration processes required for publishing and consuming data on the Web.

The rest of this paper is organized as follows: Section 2 focuses on different aspects of the systematic management of software process, and the problems that arise in the elaboration of the proposed framework. In Section 3, the elements that are part of the framework are presented, whereas Section 4 describes how its assessment has been carried out. Other research related to our proposal is described in Section 5. Finally, some conclusions and outlines of future research are collected in Section 6.

2. The business context of software process management

A business process is defined as a set of activities that are performed in coordinating an organizational and technical environment to achieve a given business objective [4]. Business Process Management (BPM) follow a cycle of continual improvement, starting with its design using some standard modeling language (e.g. BPMN), its verification and validation, its configuration, the enactment of models for its execution in a BPM engine, and its evaluation. Software processes, which are a particular type of business process, consist of a coherent set of policies, organizational structures, technologies, procedures, and the artifacts needed for designing, developing, installing and maintaining a software product [5].

The explicit definition of processes plays a key role in the major initiatives for software process improvement. For example, process modeling is included in the *Organizational Process Definition* process area of Capability Maturity Model Integration (CMMI) [6], whereas in ISO/IEC 12207, it is included in the *Process Improvement* group [7]. Modeling languages enable building process descriptions in an homogeneous way, usually by using a graphical notation. These languages help improve the appropriate understanding of the processes by all stakeholders. Software Process Modeling Language (SPML) share common elements, such as activities, resources, work products, actors and rules.

The Object Management Group (OMG) consortium published in 2002 the Software and Systems Process Engineering Metamodel Specification (SPEM), a language designed for modeling software engineering processes. It is definitely the software process definition language most commonly used [8]. The potential benefits of this language seem quite promising, such as reusing methods and processes across organizations, and laying the foundations for the automation of processes, among others [9]. However, this language has not achieved a sufficient level of acceptance in the industry, but they are only used in academic and research fields [10,11].

Unlike business processes management, controlled by the BPM systems, there are no complete suites for the definition, configuration, implementation and evaluation of software processes. However, in recent years, thanks in part to the rise of the open source movement, numerous support tools for software management and production [12] are appearing. Along with the development of these types of tools, various platforms to promote and foster cooperation between work team members in software projects and to provide support to end users have also emerged. These platforms, called software forges, are evolving towards the concept of a Application Life cycle Management (ALM) platform. The latter are designed to integrate and coordinate a number of engineering and management software tools, with the aim of covering all or most of the activities of the software life cycle.

Such support tools require, at the beginning of each new project, considerable efforts for their adaption to the specific requirements of the project and the corporate methodology. In addition, due to the slow acceptance of the SPML, the support tools, forges or ALM platforms do not often incorporate capabilities for linking with the explicit definitions or models of the processes, which causes a significant lack of consistency between the process definition and the further execution of the projects. Also, the absence of mechanisms for automation makes the modeling software process, for instance with the SPEM language, unable to offer a sufficient return on investment to be considered interesting to most companies.

In the continuous quality improvement cycle, the evaluation of processes is essential. Therefore, in order to be able to apply improvement mechanisms, it is necessary to measure and analyze the errors, deficiencies or deviations in the actual process execution. Establishing an automated measurement plan by using BPM systems is usually a relatively simple task, typically using real-time metric monitoring tools and post-mortem analysis engines, although almost real time approaches are gaining momentum [13]. In contrast, in the software engineering processes, collecting data from real projects is a complex activity because the ALM integrated platforms are not wide spread and the software process-aware support tools are in their early stages.

In addition to the analysis of metrics and indicators, technical reviews are another important set of control activities in Software Engineering. These activities are usually quite repetitive and require a significant allocation of human resources, as they are often manual activities. Reviews are usually completed at certain checkpoints throughout the software life cycle, such as at the end of certain phases, milestones, activities, or iterations (in incremental life cycles) or just before delivery to the client [14].

Despite the fact that the analysis of metrics and software revisions are essential activities to improve software quality, it is common that organizations cannot allocate sufficient effort and human resources to make this work. Therefore, research on novel mechanisms to automate technical reviews is needed.

Software management or production support tools host a large amount of information that can be used for the purpose of evaluating software processes. However, data analysis from projects deployed into software process support systems is still an emerging area in Software Engineering. One of the major problems of the integration of the information generated during the projects is the discrepancy with respect to the data models used in the different tools. Therefore, publishing data with a shared information