A Perfect Evaluation and Measurement of Software Process Development: A Systematic Literature Review

D. Prasad, 2Ahmedi Nasreen, 3S. V. Hemanth

1,2,3Dept. of CSE, Shaaz College of Engg. & Tech., Hyderabad, AP, India

Abstract

BACKGROUND—Software Process Improvement (SPI) is a systematic approach to increase the efficiency and effectiveness of a software development organization and to enhance software products. OBJECTIVE—This paper aims to identify and characterize evaluation strategies and measurements used to assess the impact of different SPI initiatives. METHOD—The systematic literature review includes 148 papers published between 1991 and 2008. The selected papers were classified according to SPI initiative, applied evaluation strategies and measurement perspectives. Potential confounding factors interfering with the evaluation of the improvement effort were assessed. RESULTS—Seven distinct evaluation strategies were identified, whereas the most common one, "Pre-Post Comparison", was applied in 49% of the inspected papers. Quality was the most measured attribute (62%), followed by Cost (41%) and Schedule (18%). Looking at measurement perspectives, "Project" represents the majority with 66%. CONCLUSION—The evaluation validity of SPI initiatives is challenged by the scarce consideration of potential confounding factors, particularly given that "Pre-Post Comparison" was identified as the most common evaluation strategy, and the inaccurate descriptions of the evaluation context. Measurements to assess the short and mid-term impact of SPI initiatives prevail, whereas long-term measurements in terms of customer satisfaction and return on investment tend to be less used.

Keywords


I. Introduction

With the increasing importance of software products in industry as well as in our every day’s life [62], the process of developing software has gained major attention by software engineering researchers and practitioners in the last three decades [93, 97-98, 106]. Software processes are human-centered activities and as such prone to unexpected or undesired performance and behaviors [44]. It is generally accepted that software processes need to be continuously assessed and improved in order to fulfill the requirements of the customers and stakeholders of the organization [44]. Software Process Improvement (SPI) encompasses the assessment and improvement of the processes and practices involved in software development [25]. SPI initiatives are henceforth referred to activities aimed at improving the software development process (see Section 3.4.3 for a definition of the different types of initiatives).

The measurement of the software process is a substantial component in the endeavor to reach predictable performance and high capability, and to ensure that process artifacts meet their specified quality requirements [41, 219]. As such, software measurement is acknowledged as essential in the improvement of software processes and products since, if the process (or the result) is not measured and evaluated, the SPI effort could address the wrong issue [52]. Software measurement is a necessary component of every SPI program or change effort, and empirical results indicate that measurement is an important factor for the initiatives’ success [33, 47]. The feedback gathered by software measurement and the evaluation of the effects of the improvement provide at least two benefits. By making the outcome visible, it motivates and justifies the effort put into the initiative. Furthermore, it enables assessment of SPI strategies and tactics [67]. However, at the same time, it is difficult to establish and implement a measurement program which provides relevant and valid information on which decisions can be based [17, 67]. There is little agreement on what should be measured, and the absence of a systematic and reliable measurement approach is regarded as a factor that contributes to the high failure rate of improvement initiatives [186]. Regardless of these problems in evaluating SPI initiatives, a plethora of evidence exists to show that improvement efforts provide the expected benefits [46, 71, 121, 137, 161, 184, 213, 224, 240, 248].

An interesting question that arises from that is how these benefits are actually assessed. A similar question was raised by Gorschek and Davis [167], where it was criticized how changes / improvements in requirements engineering processes are evaluated for their success. Inspired by the search for dependent variables [167], we conducted a Systematic Literature Review (SLR) to explore how the success of SPI initiatives is determined, and if the approach is different depending on the particular initiative. Furthermore, we investigated which types of measures are used and, based on the categorization by Gorschek and Davis [167], which perspectives (project, product or organization) are used to assess improvement initiatives. Following the idea of Evidence-Based Software Engineering (EBSE) [33] we collect and analyze knowledge from both research and practical experience. To this end we adopted the approach for conducting SLR's proposed by Kitchenham [61]. This paper is organized as follows. Background and related work is presented in Section II and our research methodology is presented in Section III. In Section IV we describe the results and answer our four major research questions. We present our conclusions in Section V.

II. Background and Related Work

A. Software Process Improvement

Software process research is motivated by the common assumption that process quality is directly related with the quality of the developed software [28, 44, 62]. The aim of software process improvement is therefore to increase product quality, but also to reduce time-to-market and production costs [28]. The mantra of many software process improvement frameworks and models originates in the Shewhart-Deming cycle [31]: establish an improvement plan, implement the new process, measure the changed process, and analyze the effect of the implemented changes [43, 49, 56, 80].

The Capability Maturity Model (CMM) [76] is an early attempt to guide organizations to increase their software development capability and process maturity [15]. Although software and process measurement is an integral part of the lower maturity
levels (repeatable and defined) and central for the managed level [75], the model only suggests concrete measurements since the diversity of project environments may evoke varying measurement needs [74]. Similarly, the Capability Maturity Model Integration (CMMI) [2, 113, 114] and ISO / IEC 15504 [35, 111] (also known as SPICE), propose various measurement mandates. The CMMI reference documentation, both for the staged and the continuous representation [113-114], provides measurement suggestions for each process area as an informative supplement to the required components of the model. The ISO / IEC 15504 standard documentation [112], on the other hand, prescribes that the process improvement has to be confirmed and defines a process measurement framework. The informative part of the ISO standard provides some rather limited exemplars of process measures without showing how the measurement framework is applied in practice.

A common characteristic of the above-mentioned improvement initiatives is their approach to identify the to-be improved processes: the actual processes are compared against a set of "best practice" processes. In case of significant divergences, improvement opportunities are identified and the elimination of the differences constitutes the actual process improvement [102]. This approach is commonly referred to as top-down [102] or prescriptive [80] improvement. In conceptual opposition to this idea is the bottom-up [102] or inductive [80] approaches to process improvement. The main principle of bottom-up improvement is a process change driven by the knowledge of the development organization and not by a set of generalized "best practices" [102]. The Quality Improvement Paradigm (QIP) / Experience Factory [7-8] is one instance in this category of improvement initiatives. As in the prescriptive approaches, measurement to control process change and to confirm goal achievement is a central part of QIP.

**B. Related work**

Gorschek and Davis present a conceptual framework for assessing the impact of requirements process changes [167]. Their central idea is that the effect of a change in the requirements process can be observed and measured at different levels: (1) Effort and quality of requirements related activities and artifacts in the requirements phase, (2) project success in terms of meeting time, budget and scope constraints, (3) product success in terms of meeting both the customer's and the company's expectations, (4) company success in terms of product portfolio and market strategies, and (5) the influence on society.

Although these concepts are described from the perspective of requirements engineering, the essence to evaluate a process change on different levels to understand its impact more thoroughly, is conveyable to software process improvement in general.

By looking at the recent literature one can find several endeavors to systematically collect and analyze the current knowledge in software measurement.

Gomez et al. [48] conducted a SLR on measurement in software engineering. The study considered in total 78 publications and tried to answer three questions: "What to measure?", "How to Measure?" and "When to Measure?" The criteria for inclusion in the review were that the publication presents current and useful measurements. To answer the first question, the study accumulated the metrics based on entities where the measures are collected and the measured attributes. The most measured entity was "Product" (79%), followed by "Project" (12%) and "Process" (9%), and the most are observed in order to increase the knowledge on these relationships. Unfortunately in many cases the context, in which the improvement is evaluated, is described unsatisfactorily (see Section 4.1.3), and an identification of confounding factors is therefore aggravated.

Generally, the effect of confounding factors on the dependent variable can be controlled by designing the study appropriately, e.g. by a random allocation of the treatment and control groups [3]. The fundamental assumption by such a design is that the confounding variables are equally distributed in each group, i.e. that the probability is high that the groups have similar properties. Therefore, if the distribution of the dependent variable is similar in both the control and treatment group, it can be concluded that the treatment has no effect.

The concept of randomization is also discussed in [64, 81, 108] in the context of software engineering experiments. Pfleeger [81], points out that the major difference between experiments and case studies is the degree of control. In order to control a potential confounding variable, the experiment can be designed in such a way that the experimental units within the distinct groups are homogeneous (blocking). Additionally, if the number of experimental units is the same in each group, the design is balanced.

Unfortunately, random sampling of projects or subjects is seldom an option in the evaluation of improvement initiatives and therefore knowing of the existence of potential confounding factors is however needed in order to be able to apply certain techniques to compensate confounding effects [3].

The matching technique, for example, leads to an evaluation design that satisfies the ceteris paribus condition by selecting groups with similar properties with respect to confounding factors [3]. By looking at the proposed solutions, several studies apply some sort of matching, e.g. by selecting similar projects in terms of size and application domain, technology or staff size (see [146, 150, 156, 161, 169, 178, 183, 190]).

There exists no systematic way to identify confounding variables [77] and as shown by the examples above,

**III. Conclusion**

This paper presents a systematic literature review that investigates how the impact of software process improvement initiatives (as defined in Section 3.4.3) is measured and evaluated. The aim is to identify and characterize the different approaches used in realistic settings, i.e. to provide a comprehensive outline and discussion of evaluation strategies and measurements used in the field to assess improvement initiatives. The major findings of this review and their implications for research are:

**A. Incomplete Context Descriptions**

Seventy-five out of 148 studies did not or only partially describe the context in which the study was carried out (see Section 3.5). In the area of process improvement it is however critical to describe the process change and its environment in order to provide results which have the potential to be reused or to be transferred into different settings. Since a considerable body of knowledge on the impact of improvement initiatives is provided by industry reports (53%, 36%), a precise and informative context description would be beneficial for both practitioners and researchers.

**B. Evaluation validity**

In more than 50% of the studies in which improvement initiatives are evaluated, "Pre-Post Comparison" is used individually or in combination with another method (see Section 4.2). Considering that confounding factors are rarely discussed (19 out of 148
studies, see Section 4.5), the accuracy of the evaluation results can be questioned. The severity of confounding is even increased by unsatisfactory context descriptions. A grounded judgment by the reader on the validity of the evaluation is prohibited by the absence of essential information.

C. Measurement Validity

Kaner and Bond [58] illustrated how important it is to define exactly the semantics of a metric and the pitfalls that arise if it is not commonly agreed what the metric actually means, i.e. which attribute it actually measures. This issue is related with further reaching questions than process improvement measurement and evaluation, and concerns fundamental problems of software measurement validity. Nevertheless, measurement definition inconsistencies, as shown in Section 4.3.2, inhibit the process of improvement itself since the comparison and communication of results is aggravated. The implication for research is that it is difficult to identify and use the appropriate measures for improvement evaluation. A better support for defining, selecting and validating measures could enable a comparable and meaningful evaluation of SPI initiatives.

D. Measurement Scope

The analysis on what is actually measured during or after an improvement initiative shows a focus on process and product quality (see Section 4.3). From the software process improvement perspective this measurement goal might be adequate and sufficient. It is however crucial to push the event horizon of improvement measurement beyond the level of projects (see Section 4.4) in order to confirm the relatively short-dated measurements at the project or product level. Since the information needs for the different stakeholders vary, appropriate improvement indicators need to be implemented. At the corporate level for example, business benefits realized by projects which encompass a wider scope than pilot improve- ment implementations are of interest. Indicators for these long-term effects can be customer satisfaction, to assess quality improvement, and return on investment to evaluate the economic benefits of improvement. The data presented in this review (see Section 4.3.2) suggests that these indicators tend to be less used in the evaluation of process improvement as other, easier to collect, indicators. The implication for research is to integrate the success indicators into a faceted view on process improvement which captures its short- and long-term impact.

E. Confounding Factors

In a majority (129, 87%) of the reviewed studies we could not identify a discussion on confounding factors that might affect the performance of SPI initiatives and thus their evaluation. Since process improvement affects many aspects of a development project, its results and effect on the organization, there are many potential such confounding factors that threaten validity. Even though study design can often be used to limit the effects it is often not practical to fully control the studied context. Thus future research on SPI should always consider and discuss confounding factors. However, we note that no good conceptual model or framework for such a discussion is currently available.

The results of this review encourage further research on the evaluation of process improvement, particularly on the conception of structured guidelines which support practitioners in the endeavor of measuring, evaluating and communicating the impact of improvement initiatives.

VI. Acknowledgment

The authors thank the anonymous reviewers whose detailed and judicious comments improved the paper considerably. This work was partially funded by the Industrial Excellence Center EASE - Embedded Applications Software Engineering (http://ease.cs.lth.se).

References


[77] J. Pearl,"Why there is no statistical test for confounding, why many think there is, and why they are almost right", Department of Statistics, UCLA, Jul. 1998.


[83] G. Premkumar, M. Potter,"Adoption of computer aided...


Mr. D. Prasad, well known Author and excellent teacher Received B.Tech(CSIT) and M.Tech(SE) from Jawaharlal Nehru Technological University is working as Associate Professor in Computer Science and Engineering and Department M.Tech Computer science engineering from Shaaz college of Engineering and Technology, He is an active member of ISTE. he has 7 years of teaching experience in various engineering colleges. To his credit couple of publications both national and international conferences /journals. His area of interest includes Data Base Management, Compiler Design, mobile computing, computer organization and other advances in computer Applications.

Miss Ahmedi Nasreen, well-developed author and faculty, pursuing M.Tech from Jawaharlal Nehru Technological University, was worked as an Assistant Professor in Dr. VRK Women’s College of Engineering and Technology for two years. Pursuing M.Tech from Shaaz College of Engineering and Technology and publishing the paper under the guidance of Mr. D. Prasad M.Tech(SE), Asscoct professor. Interested in domain areas like Database management system, Computer networks applications, Software Engineering.

Mr. S. V. Hemanth Sandepaga, well known Author and excellent teacher Received B.Tech(ECE) and M.Tech (CSE) from Jawaharlal Nehru Technological University Hyderabad is working as Associate Professor in Computer Science and Engineering and Department M.Tech Computer science engineering from Shaaz college of Engineering and Technology. He is an active member of ISTE. he has 6 years of teaching experience in various engineering colleges. To his credit couple of publications both national and international conferences /journals. His area of interest includes Data Ware and Data mining, Data Base Management, mobile computing, information security and other advances in computer Applications.