Available online at www.sciencedirect.com

ScienceDirect

The Journal of Systems and Software 81 (2008) 747–763

The Journal of Systems and Software

www.elsevier.com/locate/jss

Teaching disciplined software development

Dieter Rombach [a], Jürgen Münch [a], Alexis Ocampo [a,*], Watts S. Humphrey [b], Dan Burton [b]

[a] *Fraunhofer Institute for Experimental Software Engineering, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany*
[b] *Software Engineering Institute, Carnegie Mellon University, 15213-3890 Pittsburg, USA*

Received 1 April 2007; received in revised form 8 June 2007; accepted 9 June 2007
Available online 16 June 2007

Abstract

Discipline is an essential prerequisite for the development of large and complex software-intensive systems. However, discipline is also important on the level of individual development activities. A major challenge for teaching disciplined software development is to enable students to experience the benefits of discipline and to overcome the gap between real professional scenarios and scenarios used in software engineering university courses. Students often do not have the chance to internalize what disciplined software development means at both the individual and collaborative level. Therefore, students often feel overwhelmed by the complexity of disciplined development and later on tend to avoid applying the underlying principles. The Personal Software Process (PSP) and the Team Software Process (TSP) are tools designed to help software engineers control, manage, and improve the way they work at both the individual and collaborative level. Both tools have been considered effective means for introducing discipline into the conscience of professional developers. In this paper, we address the meaning of disciplined software development, its benefits, and the challenges of teaching it. We present a quantitative study that demonstrates the benefits of disciplined software development on the individual level and provides further experience and recommendations with PSP and TSP as teaching tools.

Keywords: Software development; Productivity; Defect density; Size estimation; Effort estimation; Yield; Personal software process; Team software process; Experimental software engineering; Software engineering education

1. Introduction

In this paper, we use a definition of discipline that relates to skill building. The "focus of discipline is on improving performance … it concerns the fidelity with which a defined process is actually followed" (Humphrey, 2006). Discipline is particularly important in software development because many software products are used in critical applications, and because undisciplined software development work has a large potential for causing economic or even physical harm. Over the last 20 years, a growing family of technical and management practices have been developed that, if properly used, will consistently deliver quality products on committed schedules. However, when team members do not properly follow these practices, their projects are typically late, over cost, and produce poor-quality products.

The Personal Software Process (PSP) course guides faculty in teaching disciplined development practices to software engineering and computer science students (Humphrey, 2005). While the PSP has not yet been widely adopted by academic programs, there is increasing industrial use and the results show that, when engineers are disciplined in their personal practices, their performance improves. This paper summarizes a study of the data gathered while training 3090 engineers. Most of the students were experienced engineers working for industrial software development organizations, and the instructors were either from the Software Engineering Institute (SEI) at Carnegie Mellon University or were trained by the SEI.

* Corresponding author. Tel.: +49 631 6800 2167; fax: +49 631 6800 1399.

E-mail addresses: Dieter.Rombach@iese.fraunhofer.de (D. Rombach), Juergen.Muench@iese.fraunhofer.de (J. Münch), Alexis.Ocampo@iese.fraunhofer.de (A. Ocampo), watts@sei.cmu.edu (W.S. Humphrey), dburton@sei.cmu.edu (D. Burton).

1.1. Why is discipline important for software organizations?

The performance of a development organization is determined by the performance of its engineering teams. Further, the performance of an engineering team is determined by the performance of the team members. Finally, the performance of the engineers is, at least in part, determined by the practices these engineers follow in doing their work.

While communication skills, native ability, intelligence, and experience have an unquestioned effect on engineering performance, this study shows that the predictability, quality, and productivity of a software developer's work can be measurably improved through training in disciplined personal practices. Furthermore, studies have shown that this improved performance at the personal level results in comparable improvements in team and project performance (Davis et al., 2003; McAndrews, 2000). These benefits are typically manifested by shorter development cycle times, fewer test defects, and reduced development and maintenance costs.

1.2. Why is discipline important for students?

Development work is becoming more challenging every year, and to succeed at this work, aspiring engineers must focus on building their personal capabilities. "Excellence starts with the individual. Achieving excellence is a constant struggle, principally because the world is changing. What was once considered excellent no longer is. This means that we must continually focus on improving our personal capabilities" (Humphrey, 2000). The critical need, then, is to understand how performance is evaluated and know what would constitute excellence.

While the performance of students is largely determined by their ability to get good grades, the technical proficiency of practicing engineers is not as significant in performance evaluations and promotions. Except for the occasional high- and low-performing exceptions, most graduate engineers are assumed to be technically competent. One of the major differentiators in industry is the engineer's ability to consistently and predictably produce quality results. When engineers follow the disciplines taught by the PSP, they can accurately plan their work, make responsible commitments, consistently meet their commitments, and produce high-quality results.

While these skills are important to engineering management, they are particularly important to practicing engineers. The reason is that when engineers consistently meet their commitments, their managers soon realize that they can manage themselves and still produce excellent results. Then, since managers are typically very busy people, they will largely trust these engineers to manage themselves, and they will continue to trust them for as long as the engineers continue meeting their commitments. Finally, as any experienced engineer will attest, the ideal engineering job is to be given an interesting and challenging assignment and to be trusted to manage the work yourself. That is the benefit of doing disciplined engineering work.

1.3. Why is discipline important for software education?

Today, a typical software education does not teach disciplined engineering practices. As a result, the most common experience in software development organizations is that their products are late, over cost, and of poor quality. This means that typical software professionals work long hours under severe schedule pressure and spend a large portion of their time fixing defective products. Few engineers like to have pizza at their desks for dinner, work on most weekends, and to stay late into the night fixing defects in test.

Most people prefer more balanced and satisfying careers. Software engineering, when done with proper discipline, can be rewarding. It involves teamwork, creating exciting and useful products, and having the satisfaction of seeing your own creations do what you intended them to do. It is potentially a great career. However, today, software engineering has a poor image and student enrolments are falling world wide, despite the demand for software professionals. To meet industry needs, and to have a growing and vibrant academic community in computer science and software engineering, the software development career must be made more attractive to potential students. This is another important reason to teach disciplined software development.

The rest of this article is structured as follows: Section 2 presents the main characteristics of disciplined software development; Section 3 presents details of the study based on data collected from 3090 engineers, who participated in PSP trainings. The study's main objective was to investigate the effects of disciplined software development on the engineer's ability to consistently and predictably produce quality results; Section 4 presents a collection of experiences and lessons learned from different institutions in the world who have used PSP and the TSP introductory course as part of their curriculum; Section 5 presents a set of recommendations for teaching disciplined software development based on the study's observations and the lessons learned from several institutions; Section 6 presents the conclusions of this article.

2. Disciplined software development

Developing software and software-intensive systems in a disciplined way requires a significant transition from craft-based to engineering-style development. While mature organizations widely aim at transitioning organizational structures and procedures towards engineering-style software development, it is also important to apply engineering principles on the level of individual developers. One of the reasons is that most methods applied in software development are significantly human-based. As a consequence, the