Category: Software & Systems Design

An Overview of Software Engineering Process and Its Improvement

Alain April
École de Technologie Supérieure, Montréal, Canada

Claude Laporte
École de Technologie Supérieure, Montréal, Canada

INTRODUCTION

The software engineering process is concerned with the definition, implementation, measurement, change, and improvement of software processes.

This short article presents software engineering process knowledge along the lines of the software engineering body of knowledge (International Organization for Standardization & International Electrotechnical Commission [ISO/IEC], 2005b). The objective of the software engineering process is to implement new or better processes in current software engineering practice.

BACKGROUND

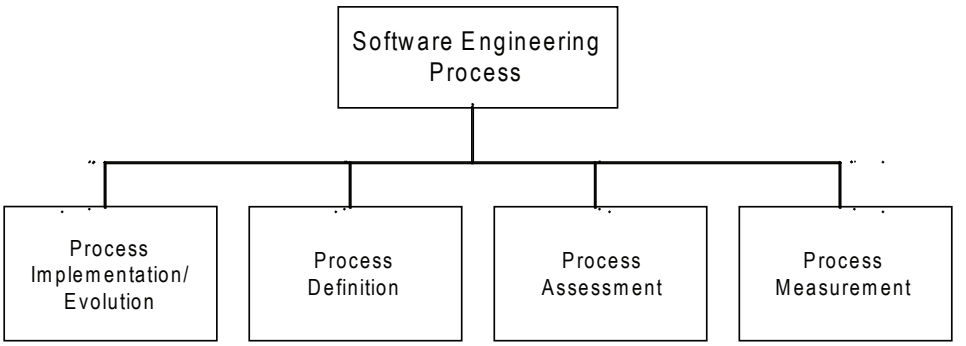
Software engineering is a young discipline, and many authors maintain that process engineering is crucial to its success, as well as being key to software quality assurance activities. This article presents generally accepted knowledge about the software engineering process. This knowledge has been adapted from industrial engineering, the management sciences, and human resources management. We have witnessed the emergence of software engineering process literature during the past 20 years and watched as some process topics have appeared while others have disappeared. This article presents four key topics (see Figure 1) that represent the fundamental concepts that must be acquired by all software engineers.

PROCESS IMPLEMENTATION AND EVOLUTION

Process implementation and change concern the initial deployment of processes and ongoing changes designed to improve and develop a supporting infrastructure (software process assets). Software engineering process activities typically follow a life cycle in which some process models are used as a reference, and certain practical considerations must be considered to ensure their success.

The first section of this article introduces concepts relating to the initial deployment of processes and to the improvement of current processes. In both cases, existing software engineering practices have to evolve. If the evolution process is extensive, then the possibility of cultural changes within

Figure 1. Key topics in the software engineering process and its improvement



the organization may need to be addressed to lower the risk of resistance and failure.

Software process improvement typically follows an improvement life cycle composed of four activities: (a) Establish the process infrastructure and assets, (b) plan the implementation (or improvement), (c) implement and evolve the process, and (d) evaluate the process. Improvement is often a project in and of itself, requiring appropriate planning, resources, monitoring, and review. Completing these life cycle activities permits continuous feedback and improvement of the software process. The first activity, establishing the process infrastructure and assets, involves establishing commitment to the process implementation and change, and acquiring the appropriate resources and personnel. The objective of the second activity, planning the implementation (or improvement), is to describe and communicate the improvement project's objectives and process needs following an assessment of the strengths and weaknesses of the current processes. The third activity, implementing and evolving the process, involves executing the planning step and deploying new processes or evolving existing processes, or both. This activity will often require piloting the new or enhanced processes. The last activity, evaluating the process, is concerned with measuring the resulting process and assessing how well it has achieved the initial objectives. This information is then used as input for subsequent improvement cycles.

The need for an appropriate software engineering infrastructure should always be considered in process improvement. This includes having the resources as well as a clear assignment of process ownership. Management commitment is essential to the success of the process improvement effort. Having an individual or an isolated group develop and evolve the software engineering processes in isolation, sometimes using proven practice handbooks, may not be the best approach as it often creates the impression that the process has been imposed by an individual or a specific organization (like quality assurance). It would be better to establish mixed-group committees that are involved in the software engineering process definition and evolution as this will ensure better representation and involvement of all software engineering staff. Two examples of such committees are the Software Engineering Process Group (Fowler & Rifkin, 1990) and the Experience Factory (Basili, Caldiera, McGarry, Pajerski, Page, & Waligora, 1992).

Moitra (1998) presents guidelines for process implementation and evolution within software engineering organizations. Hutton (1994) debates the importance of change agents in the case of major process evolution. Organizational change can also be viewed from the perspective of technology transfer (Rogers, 1983). Krasner (1999) presents examples of software definition and evolution initiatives.

PROCESS DEFINITION

The defining of processes can be represented in models, as well as in the automated process infrastructure. In an organization, a process is often composed as a procedure, a policy, or a standard, and software engineering processes are defined to harmonize software engineering activities and communication, as well as to support process improvement and its automation. In the software engineering body of knowledge, a process is defined in terms of four perspectives: life-cycle models, software life-cycle processes, notations, and automation.

Life-cycle models serve as high-level definitions of the phases that occur during development, maintenance, and operations. They are not aimed at providing detailed definitions, but rather at highlighting the key activities and their interdependencies. Examples of life-cycle models in practice are the waterfall model, the prototyping model, the evolutionary model, incremental or iterative development, the spiral model, and the reusable software model, among others (Comer, 1997).

Definitions of life-cycle processes tend to be more detailed than framework models, and unlike the standards associated with the latter, life-cycle process standards do not attempt to order their processes in time. Therefore, in principle, life-cycle processes can be arranged to fit any of the life-cycle models. The main reference in this area is ISO/IEC (1995).

Other important standards providing process definitions include the following.

- Institute of Electrical and Electronics Engineers (IEEE) Standard 1074: developing software life cycle processes (IEEE, 1991)
- ISO/IEC Standard 14764: software maintenance (ISO/IEC, 2006)
- ISO/IEC Standard 19759: software measurement process (ISO/IEC, 2005b)

To meet some certification criteria, like ISO9001 (ISO, 2000), the CMMi (capability maturity model integration; Software Engineering Institute [SEI], 2006), or the Sarbanes-Oxley Act (SOX; Securities and Exchange Commission [SEC], 2002), the definition of software processes must be compliant with quality management standards and other reference guides. ISO9001 provides requirements for quality management processes. Specifically, for the software industry, ISO/IEC 90003 (ISO, 2004) interprets each ISO9001 clause, and ISO/IEC 20000-1 (ISO/IEC, 2005a) has recently been released to address the IT service quality management system.

Processes can be defined at different levels of abstraction (Pfleeger, 2001). Various elements of a process can be