8-15-1997

Worst Practices: A Field Report on Software Development Weaknesses

Eugene G. McGuire
American University, mcguire@american.edu

Worst Practices: A Field Report on Software Development Weaknesses

Eugene G. McGuire
Computer Science and Information Systems
American University, Washington, D.C. 20016-8116
mcguire@american.edu

Introduction

One model of software process maturity that has received considerable attention is the Capability Maturity Model (CMM) developed by the Software Engineering Institute (SEI) at Carnegie Mellon University in Pittsburgh (Paulk et al , 1993). The SEI was established in the mid 1980s at Carnegie Mellon University to develop a better understanding of software design and development. The SEI, with the input of the software development community, has formulated a structured assessment methodology for evaluating software processes. Many organizations have been examined using the SEI CMM methods and the data is used to provide a view of the status of software process maturity.

Although the CMM and the quality principles it embodies are well known in the software industry there are still many challenges to overcome in successfully implementing a sustainable, quality software development culture. These challenges are evidenced by the prevalence of many worst practices which are still common in the software development environment. This paper is based on extensive research and applied field observations and briefly discusses some of these worst practices.

Software Process Improvement

Software process improvement arose out of the quality principles of Deming, Juran, and Crosby in the mid-1980s in response to the belief that better management of software development processes would lead to improved software reliability and quality. One of the better known models of software process maturity is the CMM.
Software process improvement as typified by the CMM provides guidelines for an organization to employ a structured approach to strategically improve all aspects of a software development operation and dramatically changes or extends concepts previously employed under recognized systems development models such as the waterfall and spiral approach (Royce, 1970; Boehm, 1988). Organizations previously would often attempt to improve performance in an ad-hoc manner by using technology driven approaches instead of addressing root problems. For example, organizations would often assign more staff or more tools to an already late and over-budget development project rather than addressing the reasons why the project had problems.

In contrast, in a mature organization (as measured under the CMM framework), managers use metrics to quantitatively measure the quality of a software product. Schedules and budgets are based on historical performance and are realistic; the predicted results for product cost, schedule, functionality, and quality are routinely achieved. Based on the predictability of the project results, reliable management decisions can be made about tradeoffs among these outcomes. Models such as the CMM are designed to contribute toward lower costs and reduced intervals because they specify high quality processes. These processes are capable of predictable results (producing high quality software products) which in turn help with the planning and management of projects.

The maturity framework underlying the CMM applies quality management practices to the process of software development. The framework rates organizations as being at one of five

levels of software process maturity ranging from an initial level characterized by ad-hoc practices to an optimizing level characterized by continuous improvement techniques. A major theme of the CMM's maturity framework is that a software activity can be improved if the same activity can be predictably repeated. This philosophy pertains to all the people, processes, and technologies that comprise the software activity.

The maturity levels are well-defined evolutionary plateaus that organizations strive for to improve their software processes. Each level contains Key Process Areas (KPAs) that when satisfied bring consistency to components of the software process. For example, within the Repeatable maturity level (CMM Level 2), one of the KPAs is Requirements Management. A software process that can satisfy the goals and associated activities specified by the CMM for effective requirements management has a higher level of maturity (and more predictability) than a software process than cannot satisfy the goals of this KPA.

For example, a level one organization may not have effective written policies on how to determine completeness, consistency, and clarity of requirements received from the customer and consequently such an organization would be unable to ensure consistency and repeatability in satisfying customer expectations regarding the requirements for the software project. The CMM questionnaire addresses this issue from an organizational, management, people, and process focus to insure that the activity can be first isolated for improvement and then integrated into an overall maturity model delineated into key process areas.

Organizational, Team and People Issues

Part of the reason for the recent attention to quality in software development is that empirical research consistently shows that many software development projects suffer from a lack of proven and well established methods. Models such as the CMM are only now advancing beyond the embryonic stage and being accepted by a wide variety of organizations. They still have not yet undergone the rigorous test of years of industrial application. In fact, many organizations that begin assessments, do not follow through in improving their development processes.

Many organizations, nevertheless, employ talented software engineers and managers and produce successful software products even though they are not operating with a strong process focus. In such organizations, it is appropriate to examine if team and management issues arise as those software development groups move towards a more structured, process-oriented environment.

Most improvement programs to date have emphasized process or technology and not people. People management practices in many organizations, despite a significant amount of literature addressing people-related issues (Pressman, 1995; Rasch & Tosi, 1992; Thompson & McParland, 1993), do not address people issues in a systematic and structured manner. In addition, most managers are untrained or inadequately trained in implementing corrective solutions once people problems are identified (Statz, 1994; Zahniser, 1993). Also, organizational factors, while widely cited (Constantine, 1993; McIntyre, 1992, 1994) are often not systematically analyzed for their effect on process improvement efforts.

The characteristics of the SEI levels show that information systems professionals who work on software engineering projects must be capable of being highly productive in complex, team-oriented environments with strong emphases on process control and overall quality (Walz et al, 1993; Zultner, 1993). These requirements are critical but may not always be present in current

information systems professionals. The lack of appropriate team and process control skills can greatly contribute to internal organizational volatility.

The CMM requires that organizational management, users, customers, and software development professionals communicate and collaborate frequently and effectively to achieve quality and repeatable processes. In fact, team training is specifically required to fully satisfy the requirements of some key process areas in the CMM.

The CMM breaks down the software engineering capabilities of organizations into 5 maturity levels from Level 1 (Initial) to Level 5 (Optimizing). Generally, the levels can be characterized and distinguished in terms of *team* and *management* processes as:

Level 1 -- *Initial*: There are no well-defined procedures and management. Roles of the design team members are not well defined. Project and design information is not stored or communicated effectively. The organization does not consistently apply good engineering management to the software development process. Current statistics show that over 70% of assessed organizations are still Level 1. Worst practices of these organizations include not defining roles and responsibilities and not defining development procedures in a meaningful way, i.e. with detail that is actually implemented in daily routines.

Level 2 -- *Repeatable*: The organization has achieved a stable process with a repeatable management control level by initiating rigorous project management of commitments, costs, schedules, and changes. The organization uses standard practices for managing development activities including: staff estimating; cost estimating; scheduling; change and version control on requirements, specifications, and design data; and managerial reviews. The managers are trained in these management practices and software engineers are trained on the use of tools and procedures. Worst practices at this level include adopting (or retaining) too loose an approach to training. Managers should begin building training time into project schedules at this level.

Level 3 -- *Defined*: The organization has defined the process as a basis for consistent implementation and better understanding. Goals, objectives and strategies are developed and promulgated. The software development organization is multi-disciplinary and a Software Engineering Process Group (SEPG) is established to lead process improvement. Worst practices at this level include having an SEPG composed only of part-time members. At this level, the SEPG should be institutionalized and have at least a few full-time members (unless it is a very small organization).

Level 4 -- *Managed*: The organization has initiated comprehensive process measurements and analysis. This is when the most significant quality improvements begin. The senior managers are involved in the product and process management. The performance of the design process is monitored and measured; data gathering is repeatable and auditable. The organization typically bases its operating decisions on quantitative process data, and conducts extensive analysis of the data gathered during reviews and tests. Worst practices at this level include basing the quantitative analysis and review of activities on a metrics program that is well aligned with the type of work being done. Metrics must be meaningful.

Level 5 -- *Optimizing*: The organization now has a foundation for continuously improving and optimizing the process. The organization conducts analyses of the metrics data gathered during the design process and use these for planning improvements. They focus on improving and optimizing the process and products by establishing improvement goals for quality, productivity, and cost. Worst practices at this level include not identifying meaningful improvement areas.

Requirements Management

Requirements Management is the first KPA in CMM Level 2 and represents one of the underpinnings of good project management. Nevertheless, research and field experience show that this area is still often neglected. Worst practices in this area include not establishing a good initial baseline of software requirements, not establishing a change control procedure for requirements, and not establishing and maintaining a requirements traceability matrix throughout the life cycle of the project.

A survey of over 8000 projects found that the top three reasons that projects were delivered late, over budget, and with less functionality than desired all had to do with requirements management practices: lack of user input, incomplete requirements, and changing requirements (The Standish Group, 1994). A survey of projects by the SEI reached essentially the same conclusion: more than half the projects surveyed suffered from inadequate requirements management (Kitson & Masters, 1993).

Getting a requirement right in the first place typically costs 50 to 200 times less than waiting until construction or maintenance to get it right (Boehm & Papaccio,1988).
In a study of 107 projects in 70 different organizations (Chatzoglou & Macaulay, 1995) it was found that:

1) Requirements gathering is an iterative process since in only 18% of the projects just one iteration was performed. In 32% of the of the projects two iterations occurred, while in 50% of the projects the requirements process was completed in three or more iterations.
2) The elapsed time of the requirements gathering usually represents more than 15% of the total elapsed time; however, the cost allocated to the requirements gathering process is 515% of the total cost.

3) 35% of the projects failed to capture the necessary requirements. This was caused by:
lack of time (61% of projects)
poor access to information (51% of projects)
insufficient manpower (22% of projects)
the cost (19% of projects)

Project Planning and Project Tracking

Two other KPAs in CMM Level 2 involve planning and tracking the activities, cost, and effort for the software development project. During the project planning phase it is important to use rigorous estimating algorithms and procedures based on hard, quantifiable data to derive the project schedule. During the project tracking phase it is essential to be able to accurately compare actuals to estimates in order to take corrective action when substantial deviations appear.

Worst practices encountered in the field include preparing the project plan using estimates based solely on experience. Many times these estimates are no more than best guesses from the more experienced people on the project. This is often the only method available because historical data from other similar projects has not been retained and made available for future estimating use. Worst practices in project tracking include gathering metrics on actuals that are not particularly meaningful. This can occur when the metrics collection is undertaken primarily for documenting the existence of this activity instead of for input to systematic replanning activities.

Software Quality Assurance

Another KPA at CMM Level 2 specifically addresses quality assurance activities within the software development project. These activities apply to both product and process quality and require an independent assessment of conformance to standards. It is important to emphasize the CMM and other similar process improvement models declare that overall quality is the responsibility of everyone on the project team and that quality cannot simply be built in at any particular project stage or by one individual or group. With this understanding the CMM nevertheless requires that an independent person or group who is formally responsible for software product and process quality objectively verify conformance to standards throughout the project life cycle.

Worst practices in this area include focusing software quality assurance primarily or exclusively on product testing and excluding process issues. The CMM requires that the software quality assurance person or group review or audit the process activities in all the KPAs in the model. In addition, these reviews and audits have to be performed independent of the project management structure meaning that senior management and not project management is the driver of this activity. Worst practices in the field include not effectively structuring the independent aspect of this activity.

Conclusion

Despite widespread interest in process improvement models such as the CMM, many software development organizations are still struggling with fundamental issues of people and project management as they strive to develop higher quality software. Extensive field observation shows that there is still considerable room for further maturity and improvement in these areas.

References available on request from author.