Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1999 Proceedings

Americas Conference on Information Systems (AMCIS)

December 1999

Software Team Development in the Capability Maturity Model,

Eugene McGuire
American University

Software Team Development in the Capability Maturity Model

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

Abstract

As software development organizations become more process-oriented they usually discover that they must address people and team development in an expanded and more systematic manner. The software Capability Maturity Model (CMM®) developed by the Software Engineering Institute specifies in great detail what needs in the system development life cycle; how it is done is greatly dependent on people and teams. This paper discusses team development issues for organizations to consider as they pursue higher maturity levels.

Introduction

In recent years one of the most frequently cited critical success factors for improved software development is the effective use of people, especially as they are formed into teams that are empowered to make decisions and are focused on the customer, quality, and process. There is a growing body of literature that addresses the use of people and teams in information systems in general and software engineering in particular, particularly as they are involved in complex projects (Constantine, 1995; Demarco & Lister, 1987; Glass, 1995; Humphrey, 1997; Igbaria et al, 1994; McGuire, 1997; Rash & Tosi, 1992).

Emphasized in this literature are strategic and competitive benefits of effectively using teams and teamwork in software development. These benefits include: competence transfer and build-up, improved quality, easier coordination, performance satisfaction, flexibility, progress visibility, and delivery precision (McGuire, 1996a, 1996b; McGuire & LaSalle, 1996). The role of organizational culture (Constantine, 1993) and specific organizational issues related to quality software (McIntyre, 1992; 1994) are also receiving wide attention in the software engineering literature. Growing understanding of the importance of these factors on the software design and development activities of the organization have helped lead many organizations to focus on the team and process aspects of their software development life cycle.

Software Process Improvement and Teams

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). 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). 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.

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. Most improvement programs to date, however, have emphasized process or technology and not people. People management practices in many organizations, despite a significant amount of literature addressing people-related issues [Kim & Umanath, 1993; Kraut, 1995; 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 [Rettig & Simmons, 1993; Statz, 1994; Zahniser, 1993]. Also, organizational factors, while widely cited [Constantine, 1993; 1995; 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. Often, however, organizations do not adequately prepare team members for operating effectively in team-based environments.

Two fundamental issues generally hinder teams in adopting the CMM®:

- Failure to understand the difficulty of the deployment of the concepts. Because the concepts seem simple and logical, many people think that the implementation will also be simple or will just happen.
- What seems simple, straightforward and easy in concept may cause an organization, team, or individual to think they are already doing it.

In addition, the CMM® is based on a continuous improvement philosophy. A continuous improvement focus concerning team responsibilities should address:

- how well the team's routines support effective decision making (should be as close as possible to the line of action)
- how effectively the team's work is coordinated (should be coordinated so team can make seamless and timely hand-offs)
- how effectively existing human resource systems (such as team member selection, training and reward) support the goals of the team (should support the level of ownership and teamwork necessary for the team to achieve its goals and maintain a high quality of worklife)
- how well existing values and norms support the team's goals and desired outcomes.

Team Development in the CMM®

Effective teams of all types demonstrate similar characteristics including: clear sense of purpose, informal climate, participation, listening, constructive disagreement, consensus, open communication, clear roles and work assignments, shared leadership, external relations, style diversity, and self-assessment (McGregor, 1960). These characteristics remain appropriate goals and evaluation criteria for teams working in a CMM® environment.

In addition, successful and effective teams in a CMM® environment generally establish team norms or routines that have been defined by the team members, maintain team focus and provide a common view of how to do the work, and are subject to continuous improvement (McGuire, 1996a). Team activities (separate from project or assignment activities) should address the following team responsibilities:

- planning and distributing work
- building commitment around the plan and the goals
- following the plan and the process
- reporting progress and deviations and initiating replanning
- reviewing produced material and following up on results of the review
- ensuring team competence and continuous improvement
- maintaining weekly schedules and keeping action lists

In addition the team should identify:

- the workflow needed to meet its goals most effectively and efficiently
- the information flow needed to support the work flow and facilitate decision making
- the process measures it will use to verify that its goals are being met
- how to perform essential technical tasks (including operational and troubleshooting tasks)
- how to gather and interpret information about customer requirements and external demands, conformance to end- and interim-product measures, and system performance
- how to make data-based decisions, solve problems, and negotiate resolutions as a group
- how to plan, contribute to, and evaluate team meetings

Warning signs that teams are not operating effectively in a CMM®-based environment include:

- Too many action items for proper focus.
- No measurement of baseline data before improvement.
- No measurement of improvement.
- Measurement is not based on proper data.
- Failure to document activities: actions, minutes, etc.
- Reacting to problems vs. preventing them.
- Jumping to action before analyzing the problem.
- Failure to communicate progress and action to team members.
- Failure to deal with team dynamics, cultural problems, openness and change.
- Failure to concentrate on processes under the team's control of influence.
- Improper follow-up to support the actions implemented causing the solution to become ineffective.
- Failure to schedule time for action items.

When teams are found to be deficient in the above they may require additional training to reinforce the philosophy and principles necessary to successfully operate in a CMM®-based software development environment. This training should include developing:

- a software development process that provides the basis for executing projects of various sizes and types, adapting standard processes based on the process drivers for a given organization;
- the skills for tailoring a valid process to a life-cycle model for a specific project;
- the skills to identify risks to a project, assess their impact, build mitigation plans, and monitor status;
- knowledge of effective project planning, estimating, and scheduling techniques;
- knowledge of effective techniques for monitoring and managing a software project;
- an understanding that quality results are defined by customer needs;
- knowledge that quality results are created by project teams that care about their customer's success;
- a project team that is concerned about its ability to do its job well;
- an understanding of how to leverage lessons learned from completed projects to improve future projects.

CMM®-Based Environments

The information in the preceding section can be categorized into the following areas that are critical for successful CMM® environments:

Process Focus: Organizations with a process focus work to prevent crises from occurring instead of reacting to them after they occur. User satisfaction is actively monitored and the quality of the process is quantitatively measured so all aspects of the process can be continuously improved. As the focus on process and process maturity increases, institutionalization of organizational processes is achieved via policies, standards, and organizational structure. Individual approaches to problem solving are integrated into the process focus rather than used instead of a process focus.

Quality Focus: One of the most important issues affecting the software development profession is that of quality. Over the past twenty years there have been thousands of articles addressing the issues of quality. However, views about quality have been shaped to a considerable degree by five major writers: Crosby, Deming, Feigenbaum, Juran, and Taguchi. Crosby's (1979) views of quality may be expressed by three concepts: the notion of "zero defects;" the definition of quality as "conformance to requirements;" and the view that quality is assured by a highly structured, step-by-step program focused on improving quality.

Team Orientation: This is perhaps the most commonly cited characteristic of successful process and quality environments. Ramifications of this structural change includes requiring that employees not adopt or retain insulated work modes but instead develop business, managerial, and political skills to successfully negotiate with multiple constituencies and integrate their work and decision making with the strategic plans of the organization. There is often a considerable emphasis placed on training and education in the transition phase to a team-based organization. Empowerment of teams requires that they receive direction and have the capability to make a difference in the attainment of goals.

Change Agentry: Change management strategies must be utilized from the beginning of a process improvement effort. Benefits of the change must be clearly articulated and reward systems implemented that recognize the shift from individual effort to team- and process-driven effort. Barriers that inhibit the change from successfully occurring must be removed. Management may find it appropriate to adopt a coaching and facilitating style instead of a command and control style in dealing with employees when change is occurring. If so, management needs to be trained in this area.

Organizational Learning: Many organizations are realizing that continual learning patterns must become institutionalized for that organization to achieve maximum use of people and remain competitive. These organizations work hard at facilitating systems thinking (seeing the organization as a whole); enhancing existing mental models of existing processes and procedures (thinking outside the box); and team learning (where collaboration and group success are the norms).

Communication: Software professionals must communicate in a variety of settings using oral, written, and multimedia techniques. Change agent roles, in particular, require the ability to effectively articulate both strategic and tactical planning to multiple levels of an organization during the life cycle of a project or program.

Problem Solving: Software professionals must be able to choose from a variety of different problem solving methodologies to analytically formulate a solution. They must think creatively in solving problems and be able to work on project teams and use group methods to define and solve problems.

Conclusions

As more organizations adopt the CMM® as their standard for software development, there will be a correspondingly increased focus on how to most effectively utilize and reward teams of software development professionals. The CMM® is highly descriptive of the processes involved in producing high-quality software but silent on the desired behaviors required of software professionals who are responsible for this activity. This paper discusses some of those desired behaviors and suggests some specific types of training for these professionals if those behaviors are absent or weak. Establishing and reinforcing these desired behaviors provides the basis for a new performance and evaluation system as well as the foundation for a new organizational culture of quality and process-driven software engineering as described in greater detail in recent literature (Weinberg, 1994; Weigers, 1996; Hohmann, 1997).

References

References available upon request.