Communications of the IIMA

Volume 6 | Issue 3 Article 3

2006

A Case Study of CMM Deployment at SBC Communications

James T. Sofos
SBC Communications

Jack T. Marchewka
Northern Illinois University

A Case Study of CMM Deployment at SBC Communications

James T. Sofos
SBC Communications

Jack T. Marchewka
Department of Operations Management & Information Systems
Northern Illinois University
DeKalb, Illinois 60115
815-753-1332
jmarchewka@niu.edu

ABSTRACT

*Recently, SBC Communications (SBC) initiated a project to improve the software development processes of its Information Technology (IT) group. The project was named EXPRESS (**E**xcellence in **P**rocess for **E**nterprise **S**oftware **S**olutions), and the goal was to transform the IT function to a Capability Maturity Model (CMM) Level 3 operation. SBC's management hopes that achieving CMM level 3 will result in improved delivery of IT services and products through greater efficiency by having a standardized means to initiate and perform IT work. This paper provides a case study that describes the EXPRESS project. This will include comparing SBC's former project management strategy with the goals of the new CMM strategy and how CMM Level 3 and EXPRESS are being implemented at SBC. Several lessons learned that resulted from the rollout of the EXPRESS project are discussed as well.*

AN OVERVIEW OF THE CAPABILITY MATURITY MODEL INTEGRATION (CMMI)

In 1986, the Software Engineering Institute (SEI), a federally funded research development center at Carnegie Mellon University, set out to help organizations improve their software development processes. With the help of the Mitre Corporation and Watts Humphrey, a framework was developed to assess and evaluate the capability of software processes and their maturity. This work evolved into the Capability Maturity Model (CMM) (Humphrey, 1988). The CMM for Software version 1.0 was published in 1991 and provided an evolutionary path for organizations to improve their underlying software processes. Two years later, the CMM was revised as version 1.1 with another revision planned for in 1997 as version 2.0. This planned version was never released but it did serve as a basis for the Capability Maturity Model Integration (CMMI) initiative.

Since the original CMM initiative in 1991, organizations have used a number of CMMs for different disciplines or areas. Although useful, using several different models can be problematic because a particular model may limit process improvements to a specific area or discipline within the organization. Often these process improvements are not limited to a specific area but cut across different disciplines. As a result, the CMMI project was initiated to sort out the problem of using a number of CMMs (Chrissis, Konrad, Scrum, 2003). Currently, CMMI combined three models: The Capability Maturity Model for Software (SW-CMM), the System Engineering Capability Model (SECM), and the Integrated Product Development Capability Maturity Model (IPD-CMM). The intent of CMMI was to combine these models into a single framework that could be used to improve processes across the organization. The CMMI framework was designed so that other disciplines could be integrated in the future.

The CMMI provides a set of recommended practices that define key process areas specific to software development. The objective of the CMMI is to provide guidance as to how an organization can best control its processes for developing and maintaining software. In addition, the CMMI provides a path for helping organizations evolve their current software processes toward software engineering and management excellence (Paulk, Curtis, et al. 1993).

To understand how the CMM may support an organization, several concepts must first be defined:

- **Software Process** - A set of activities, methods, or practices and transformations used by people to develop and maintain software and the deliverables associated with software projects. Included are such things as project plans, design documents, code, test cases, user manuals, and so forth.
- **Software Process Capability** - The expected results that can be achieved by following a particular software process. More specifically, the capability of an organization's software processes provides a way of predicting the outcomes that can be expected if the same software processes are used from one software project to the next.
- **Software Process Performance** - The actual results that are achieved by following a particular software process. Therefore, the actual results achieved through software process performance can be compared to the expected results achieved through software process capability.
- **Software Process Maturity** -The extent to which a particular software process is explicitly and consistently defined, managed, measured, controlled, and effectively used throughout the organization.

One of the keys to the CMM is using the idea of software process maturity to describe the difference between immature and mature software organizations. In an immature software organization, software processes are improvised or developed ad hoc. For example, a software project team may be faced with the task of defining user requirements. When it comes time to complete this task, the various members of the team may have different ideas concerning how to accomplish it. Several of the members may approach the task differently and, subsequently achieve different results. Even if a well-defined process that specifies the steps, tools, resources, and deliverables required is in place, the team may not follow the specified process very closely or at all.

The immature software organization is characterized as being reactive; the project manager and project team spend a great deal of their time reacting to crises or find themselves in a perpetual state of fire fighting. Schedules and budgets are usually exceeded. As a result, the quality and functionality of the software system and the associated project deliverables are often compromised. Project success is determined largely by who is (or who is not) part of the project team. In addition, immature software organizations generally do not have a way of judging or predicting quality. Since these organizations operate in a perpetual crisis mode, there never seems to be enough time to address problem issues or improve the current processes.

Mature software organizations, on the other hand, provide a stark contrast to the immature software organization. More specifically, software processes and the roles of individuals are defined explicitly and communicated throughout the organization. The software processes are consistent throughout the organization and continually improved based on experimentation or experiences. The quality of each software process is monitored so that the products and processes are predictable across different projects. Budgets and schedules are based on past projects so they are more realistic and the project goals and objectives are more likely to be achieved. Mature software organizations are proactive and they are able to follow a set of disciplined processes throughout the software project.

The CMM defines five levels of process maturity, each requiring many small steps as a path of incremental and continuous process improvement. These stages are based on many of the quality concepts and philosophies of Shewhart, Deming, Juran, and Crosby (Paulk, Curtis et al. 1993). Figure 1 illustrates the CMM framework for software process maturity. These levels allow an organization to assess its current level of software process maturity and then help it prioritize the improvement efforts it needs to reach the next higher level (Caputo 1998).

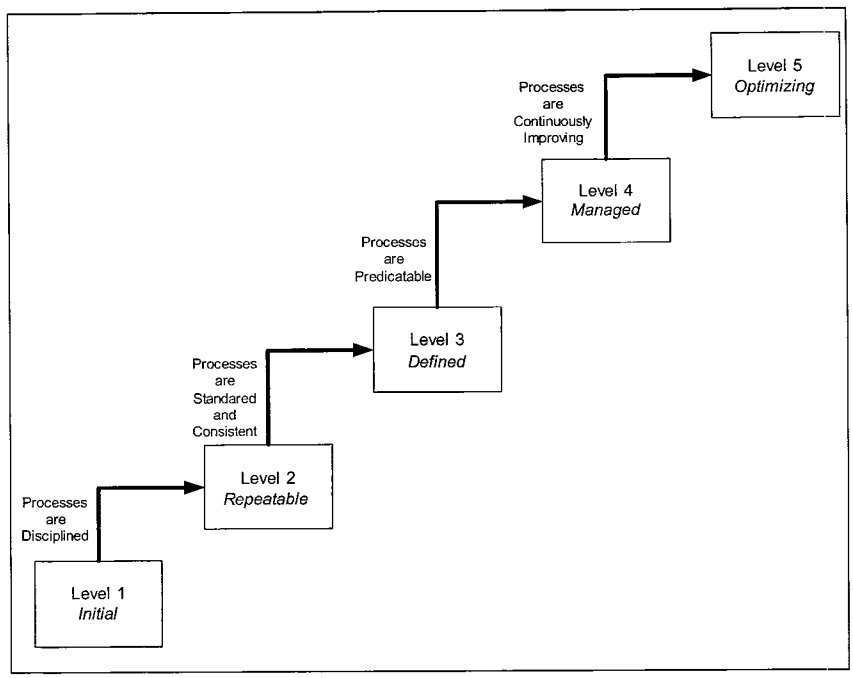

Figure 1: CMMI Levels of Software Process Maturity

Maturity levels provide a well-defined, evolutionary path for achieving a mature software process organization. With the exception of Level 1, each maturity level encompasses several key process areas that an organization must have in place in order to achieve a particular level of maturity. There are five levels of software process maturity.

Maturity Level 1 is the initial level and generally provides a starting point for many software organizations. This level is characterized by an immature software organization in which the software process is ad hoc and often reactive to crises. Few, if any, processes for developing and maintaining software are defined. The Level 1 software organization does not have a stable environment for software projects, and success of a project rests largely with the people on the project and not the processes that they follow. As a result, success is difficult to repeat across different projects throughout the organization.

Maturity Level 2 is the Repeatable Level according to CMM and Managed Level according to CMMI. At this level, basic policies, processes, and controls for managing a software project are in place. Project schedules and budgets are more likely to be realistic because planning and managing new projects is based upon past experiences with similar projects. Although software processes between projects may be different at this level, the process capability of Level 2 organizations is more disciplined because software processes are more documented, enforced, and improving. As a result, many previous project successes can be repeated by other project teams on other projects. According to CMMI, a Level 2 organization will retain all practices during times of stress. The status and delivery of services and work products are visible to management and to the other stakeholders. Work products are reviewed with stakeholders and changes are controlled.

Maturity Level 3 is the Defined level in both CMM and CMMI. At Level 3, software engineering and management processes are documented and standardized throughout the organization and become the organization's standard process. Moreover, a group is generally established to oversee the organization's software processes and an organization-wide training program to support the standard process is implemented. Thus, activities, roles, and responsibilities are well-defined and understood throughout the organization. The software process capability of this level is characterized as being standard, consistent, stable, and repeatable. However, this standard software process may be tailored to suit the individual characteristics or needs of an individual project. At the Defined Level, the standardized processes that may have been in place for Level 2 in different forms are standardized. Individual processes may be tailored to create a defined software process. This means that there are readiness criteria, inputs,

standards and procedures, verification methods (peer reviews), outputs, and completion criteria. CMMI also outlines that standard processes must be established and improved over time. Processes defined in Level 3 are more detailed and rigorous than at Level 2. The processes are managed proactively with an understanding of the interrelationships of the process activities.

Maturity Level 4 is the Managed Level according to CMM and the Quantitatively Managed Level in CMMI. For both CMM and CMMI the results of projects are more quantitatively predictable in Level 4. At this level, quantitative metrics for measuring and assessing productivity and quality are established for both software products and processes. This information is collected and stored in an organization-wide repository that can be used to analyze and evaluate software processes and products. Control over projects is achieved by reducing the variability of project performance so that it falls within acceptable control boundaries. The software processes of software organizations at this level are characterized as being quantifiable and predictable because quantitative controls are in place to determine whether the process performs within operational limits. Moreover, these controls allow for predicting trends and identifying when assignable causes occur that require immediate attention. The organization generally establishes quantitative quality goals and then measures these for software process activated across all projects. Projects then work to narrow their variation in performance. The organization can then distinguish variations that are meaningful and act to correct these. An overall software process database is used to collect and analyze the data from all projects.

Maturity Level 5 is the Optimizing Level in both CMM and CMMI. At this highest level of software process maturity, the whole organization is focused on continuous process improvement. These improvements come about as a result of innovations using new technology and methods and incremental process improvement. Moreover, the organization has the ability to identify its areas of strengths and weaknesses. Innovations and best practices based on lessons learned are identified and disseminated throughout the organization. Both CMM and CMMI provide a road map in terms of goals to institute an environment of continuous improvement. CMM/CMMI both provide key attributes of an organization that would be performing at a certain level.

As an organization's software process maturity increases, the difference between expected results and actual results narrows. In addition, performance can be expected to improve when maturity levels increase because costs and development time will decrease, while quality and productivity increase. According to the SEI, skipping maturity levels is counter-productive. If an organization was evaluated at Level 1, for example, and wanted to skip to Level 3 or Level 4, it may be difficult because the CMM identifies levels through which an organization must evolve in order to establish a culture and experiences.

The benefits of CMM and CMMI in general are that they are specifically geared toward software development whereas Six Sigma, or ISO 9000 are more manufacturing and engineering based. This, however, does not restrict a firm to using only one methodology or process. CMM/CMMI does not address certain IT operations issues, such as change and configuration management, capacity planning, help desk functions or security. CMM/CMMI sets forth goals, but does not provide specifics on how to accomplish these goals.

For example, JP Morgan Chase's IT department uses CMM, even while the firm as a whole focuses on Six Sigma principles. The firm moved from a Level 1 to Level 2 CMM IT organization and noticed a 20-25% reduction in post-implementation defects, including a 60% decreases in emergency releases to fix bugs. Suppliers are delivering better performance because of improved requirements, and projects are easier to manage because of standardized requirements and terminology (Anthes, 2004).

Some firms, though, realize that moving all the way to Level 5 of CMM/CMMI may be too costly or may not provide enough additional benefits. This is especially true of firms whose principle business is not software development. For Allstate, management believes there is a payback in moving their organization to a Level 2 or Level 3 shop. Improvements in speed to market and increased efficiencies are likely to be gained. Beyond this level though, management is unsure of the payback. Furthermore, it is difficult to find individuals with appropriate experience as implementers or assessors at those levels (Anthes, 2004).

RESEARCH METHODOLOGY

In general, a case study provides an appropriate research strategy when "how" or "why" questions are of primary interest, when the researcher does not have significant control over events, or when the phenomenon of interest takes place within some real-life context (Yin, 1994). Moreover, descriptive or qualitative research may be undertaken

when description and explanation are of greater interest than prediction (Merriam, 1988). The majority of data in this study was gathered from two interviews with the Assistant Vice President-ITS Process & Tools at SBC.

SBC Communications

SBC has a long history of providing telecommunications services throughout the United States. Over the past decade, industry forces have encouraged the old Southwestern Bell to merge with several regional communications providers. All of these were at one time part of the seven regional holding companies that were divested by the American Telephone and Telegraph Company (AT&T) in October 1983. SBC completed its acquisition of Pacific Telesis in 1997. The acquisition of Southern New England Telecom (SNET) in was completed in 1998. And finally, in 1999, SBC completed the merger with Ameritech. Despite having similar roots, over time all of these firms changed and adopted their own processes.

This history of mergers led to a wide range of practices within the firm's IT practice. Even though the organization was operating as one, the old Ameritech, PB, SNET, and SWBT regions had retained many of their old ways of doing business and few centralized or institutionalized processes existed. Often managers were left to either their own project management procedures or, over time, regional processes that had evolved under particular directors or vice presidents. In many cases, groups did not have any defined processes at all. Or, if there were any processes to be followed, they tended to be siloed within the business units. Even if there were processes within some IT groups, there were few auditing or reporting mechanisms to ensure that the goals of having a process were being met or that the process was being followed. Different cultures that had formed at the regional firms only increased the difficulty in tracking and measuring work. Similar in other parts of the business, IT had to recognize the differences and attempt build a standard culture to capitalize on the supposed economies of scale that existed as a result of the mergers.

Subsequently, this often created issues within the IT and the business communities at SBC as the business side clamored for a standard way to request work. IT and business leaders both wanted a way to determine that the work being done made sense and fit with the overall strategic goals of the firm. The business and their corresponding IT unit(s) often had set up "sweetheart" deals where work requests were accepted without any external assessment to ensure the work was sensible and aligned with the goals. IT management often did not have an idea as to what work was being performed or how it was being accomplished.

The lack of a standard process also allowed the overall quality of IT applications to suffer. With each group or team running their own quality initiatives (or often not running any quality initiative at all), there was no standard or defined way to be sure that quality was built into new or sustaining work. Furthermore, there was no assessment or auditing of what was being put into production.

EXPRESS (Excellence in Process for Enterprise Software Solutions)

Subsequently, management saw the need to address SBC's software development approach within its IT operation. The top management at SBC's Information Technology group desired a process that would allow them to be sure the "right" work was being done and in a way that brought quality and value to the business. They also wanted to reduce costs associated with the development life cycle and provide more products and services to customers more quickly.

In general, IT work thought of as a triangle with each side of the triangle representing a particular facet of the project - i.e., scope, schedule, and budget. SBC was looking for way to reduce the schedule and budget sides of the triangle while being able to meet the right scope (requirements) that would aid in meeting the organization's strategic goals. Furthermore, SBC was searching for a way to improve overall software quality. It was believed that a standard way of doing business could help accomplish this. Another potential benefit of a standard process would be the ability to more accurately know the status of individual projects and have more fungibility of resources as common templates terminology and processes were applied across all application development.

According to the Assistant Vice President-ITS Process & Tools at SBC, of all the potential process frameworks, the organization had the most in house experience and knowledge with CMM. Furthermore, CMM or other SEI standards were viewed as the standard in software development processes. From a development arena, CMM made sense within the scope of SBC's IT organization.

Since CMM does not address all aspects of software development, it was important to wrap the basics with additional process that were more applicable to SBC and the activities that occur within the IT organization. EXPRESS took those other processes, together with CMM Level 3 key attributes and rolled them into a single program that could cover all of the IT activities within SBC.

More specifically, the EXPRESS process is composed of four parts. The first part entails the project path and focuses on the identifying project type based on the expected complexity of the work. Based on the project's complexity, a different set of activities and deliverables are then required. There are three paths defined in EXPRESS, High Complexity, Low Complexity, and Small Work (database table changes, etc.).

Next, EXPRESS defines the phases and purposes, exit criteria, owners, tasks, participants, inputs, outputs, and recurring activities for each phase. Consultation and Assessment is the first EXPRESS phase. This is where an initial understanding of the problem is gathered and an impact analysis is conducted. At this point it is also determined if the work request has enough priority to proceed. If a work request successfully meets the exit criteria for the Consultation and Assessment phase, it will go into the Definition and Funding phase. In this phase, the detailed requirements are discovered, a solution approach is developed, and a high level design is completed. Funding is also secured and the deployment commitment is communicated. After completion of specific exit criteria, the work request enters the Development and Testing phase. The detailed technical design is created, the software products are built and tested, support documentation is then updated and preparations are made for deployment. Following this phase, the work requests enter the Deployment phase. In addition to the actual deployment of the solution, Key Learnings meetings are conducted and the product is transitioned from the development team to an operations support team. The client will begin to track the benefits of the application at this point.

Next, EXPRESS defines a process flow for each of these phases. Activities that support the exit criteria are organized and numbered. These activities are associated with the Key Process Areas for CMM Level 3 (i.e., peer reviews, intergroup coordination, software product engineering, integrated software management, training programs, organization process definition, and organization process focus). Finally, EXPRESS defines a set of Recurring Activities that include key process areas such as software project planning and tracking, software configuration management, software quality assurance, software subcontract management, and training. For each of these areas, EXPRESS defines checklists, roles, responsibilities, and activities to support the development of the software products.

In tandem with the EXPRESS process, tools were introduced to ease the management of software development work at SBC. An in-house intranet based tool, EXPRESS Works, was developed to control and manage the disposition of new work requests to the IT organization. Process flows and contact information were built into the tool to automate how requests are processed and assessed by the IT organization. In addition, metrics and other data can be stored for review by management. EXPRESS Works ensures that the right resources are involved in the evaluation and approval of new IT work from both a technical and project management standpoint. This forces the work to follow a defined path and cuts out any backdoors that were previous available to business units requesting work.

Document management was also implemented through the introduction of a Panagon-based document library system to manage all documents created by EXPRESS projects. Along with the repository and versioning tools that Panagon offers, a common set of naming conventions allows management to more readily verify and audit the progress of projects. By using a single document management forum, backup and replication of design, budget, requirements, and other important documents is centrally managed to reduce the risks of losing data due to unsafe document management processes.

Other tools were also standardized across the IT function so that management views a single set of metrics across all the IT organizations. More specifically, a time reporting tool was consolidated into a single tool, while a single cost and project management tool had been chosen in the years preceding EXPRESS. Currently, there is an effort under way to consolidate other tools in the reporting of project metrics. Time entry, work entry, portfolio and project management will eventually fall under an SAP-based system.

After an extensive training period, many IT teams within SBC began to utilize the EXPRESS process in the third quarter of 2004. In addition to meeting CMM Level 3 Key Process activities, EXPRESS prescribed new ways for the business groups to request work from IT, new means to control and manage project and documentation, a new

set of terminology common across all projects, and increased auditing of project teams to ensure that the right work was being undertaken in the right way. After an initial shockwave of issues and concerns brought forth by IT and business stakeholders, the process is just beginning to work. As a result of continued training, communications, and modifications to the initial process, the number of requests for help to the Software Engineering Process Group (SEPG) has been declining.

The SEPG is currently studying how to incorporate CMMI into the EXPRESS process. As the process becomes more diffused within the organization, suggestions and ideas are flowing back to the SEPG as to what is and is not working. Software Quality Assurance (SQA) audits have indicated that, so far, data are trending towards the goals of fewer defects, quicker speed to market, and reduced costs. It appears that in 2005 and 2006 the process will be institutionalized across the IT and client communities at SBC.

CONCLUSIONS AND LESSONS LEARNED

As a result of what has been the largest CMM rollout ever, several important lessons have been learned. These can be classified into two distinct categories - direct and indirect lessons learned.

Directly, the SEPG team learned that the initial assessment of the effort needed to develop the program was underestimated. It was a significant challenge to develop materials and deploy training to all the personnel involved. Having an organization spread throughout the United States compounded these issues and the logistics of the training became problematic. Given an aggressive timeline to rollout the process, the team would have been more capable with additional personnel and budgetary resources.

Another lesson learned as a result of the adoption of the process is that the teams, upon initial inspection had some sort of process, did not necessarily follow this process rigorously. This is the result of the years without a formal process or lack of process training and standardization across the organizations technical directors.

As with any major shift in process or workflows, there is often some organizational resistance to change. Of course, the deployment of a new IT development process affects areas like governance, finance, and operations outside of the directly impacted groups. The SEPG enlisted top leadership up to the Chief Technology Officer (CTO) level. SBC's CTO, was involved in town hall style meetings at several locations with large number of IT employees. This high level sponsorship impressed the importance of the program to the employees.

Indirectly, the deployment of the new IT process uncovered some issues with how work was being accomplished. Specifically this focused on the relationships between the business and individual IT groups. With the adoption of the EXPRESS process, a specified workflow ensures that work is properly prioritized. The result of this has been a large increase in consultation requests whereby the business first asks IT to conduct a feasibility study. Better reporting is the main driver of this increase in consultations. Prior to EXPRESS, there was no formal way for management to determine the costs or amount of effort required for these consultation-related activities. Currently, data is being gathered, and management will eventually be able to analyze this data to determine if any trends concerning the source of work exist.

However, the more rigorous process had initially created some consternation on the part of the business partners. As the relationships that had formed over time were broken down, some of the business partners felt left out of the process. There was some difficulty in understanding "what's a business model issue" versus "a process issue" and how to prioritize these. For years, some in the business community had desired a single way to get work requested from IT. Of course, now that this has been implemented, the backdoors to getting projects and work completed have been cut off. Some of the client community is upset by the increased attention paid to what work is occurring. In the end, this more formal process, allows both the business and IT to ensure that the right work is being done.

REFERENCES

Anthes, G.H. (2004). Quality model mania. *Computerworld.* 10. 41-45.

Caputo, K. (1988). *CMM implementation guide: Choreographing Software Process Integration and Product Improvement.* Reading, MA: Addison-Wesley.

Chrissis, M.B, Konrad M., and Shrum, S. (2003). *CMMI®: Guidelines for process integration and product improvement.* Addison Wesley Professional. Series: The SEI Series in Software Engineering. ISBN: 0321154967

Humphrey, W. (1998). Characterizing the software process: A maturity framework. *IEEE Software.* 5(3). 73-79.

Merriam, S.B. (1988). *Case Study Research in Education: A Qualitative Approach.* San Francisco: Jossey-Bass, Inc.

Paulk, M.C., Curtis, et al. (1993). The capability maturity model for software. *IEEE Software.* 10(4). 18-27.

Yin, R.K. (1994). Case Study Research: Design and Methods. Second Edition. Thousand Oaks: Sage Publications