A Field Guide to Effective Requirements Management Under SEI's Capability Maturity Model

by Dean A. Leffingwell

Abstract

The Capability Maturity Model, or CMM has emerged as the de-facto industry standard for a comprehensive software quality process. Deeper and more comprehensive than ISO9000 standards, the CMM provides a pragmatic, yet disciplined view of software activity which is being widely adopted by the commercial software marketplace. Central to the CMM is the issue of requirements management. Requirements management is a key process activity designed to ensure that software products meet the needs of the customer in both functionality and quality. This article summarized requirements management under the CMM, and provides useful guidelines for managing requirements consistent with the CMM.

Background - the SEI's Capability Maturity Model

In November of 1986, the Software Engineering Institute (SEI), located at Carnegie Mellon University, began developing a process maturity framework to help developers improve their software process[1] . In September of 1987, the SEI released a brief description of the process maturity framework which was later amplified in Watts Humphrey's book, *Managing the Software Process*.[2] By 1991, this framework evolved into what has become known as version 1.0 of the Capability Maturity Model, or CMM. In 1994, version 1.1 of the CMM was released.[3]

Version 1.1 of the CMM defines five levels of software maturity for an organization and provides a framework for moving from one level to the next (figure 1). The CMM guides developers through activities designed to help an organization improve their software process with the goal of achieving repeatability, controllability and measurability. The CMM has gained considerable credibility in software intensive industries. Adherence to the CMM and corresponding improvements in software quality have significantly lowered the cost of application development within large commercial companies.

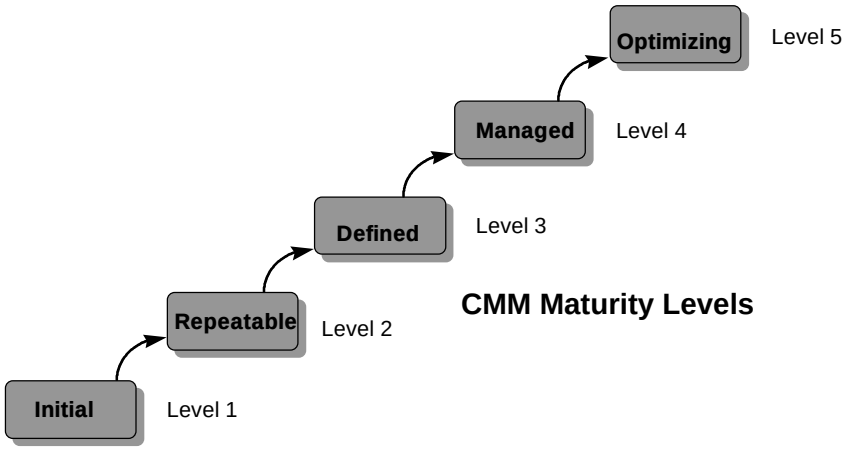

Figure 1- The CMM's Five Capability Maturity Levels

The CMM provides a framework for process improvement which consists of "key process areas." These are organizational activities that influence various aspects of the development process and resultant software quality. Table 1 illustrates the key process areas of each of the five levels of the CMM. Not surprisingly, Table 1 shows that the first key process area that must be addressed to move from the Level 1 to the Level 2 is requirements management

TABLE 1
Levels of the CMM With Key Process Areas

Level	Key Process Areas
1. **Initial** Ad hoc, even chaotic; success depends solely on individual heroics and efforts.	not applicable
2. **Repeatable** Basic project management to track functionality of application, and cost and schedule of project	Requirements Management Software Project Planning Software Project Tracking and Oversight Software Subcontract Management Software Quality Assurance Software Configuration Management
3. **Defined** The process for management and engineering is documented, standardized, and integrated, All projects use an approved, tailored version of the process.	Organization Process Focus Organization Process Definition Training Program Integrated Software Management Software Product Engineering Intergroup Coordination Peer Reviews
4. **Managed** Detailed measures of the software process and software quality metrics are collected. Both process and software products are understood and controlled.	Quantitative Process Management Software Quality Management
5. **Optimizing** Continuous process improvement is enabled by use of metrics and from piloting innovative ideas and technologies	Defect Prevention Technology Change Management Process Change Management

Return on Investment

The CMM has been in use by many organizations long enough so that meaningful return on investment statistics are appearing. Organizations like Raytheon[4] have proven that improving their development processes by moving from CMM Level 1 to Level 3 (discussed later), can lower their cost of rework by up to 50-60%. Still more dramatically, Lawrence Putnam, founder and CEO of Quantitative Software Management, which has been in the business of measuring productivity of software development since 1978, reports payoffs in the range of 70-100%/year.[5] These payoffs provide results in productivity and corresponding reduction in time to market by as much as a factor of 2-3 times! In an era of increasingly competitive environments, these improvements to the bottom line cannot be ignored by even the most jaded of organizations.

CMM Level 3 organizations have achieved productivity gains of from 200-300%

Getting Started - Requirements Management under CMM Level II

Requirements Management is the first key process area addressed by the CMM and is integral part of the software development activity.

> ***The purpose of requirements management is to establish a common understanding between the customer and the software team of the customer's requirements.***

This common understanding serves as the basis of agreement between the customer and the project team, and as such, is the central document which defines and controls the activity to follow. Requirements are controlled to establish a baseline for software engineering management use. Throughout the CMM, guidelines specify that all activities plans, schedules, and software work products are to be developed and modified as necessary so as to be consistent with the requirements which are allocated to software. In this manner, the CMM moves the organization towards an integrated view - wherein both technical requirements and project plans and activities must be kept consistent with each other. To support this process, software requirements must be documented and reviewed by software managers and affected groups including representatives of the customer and user community.

Software requirements must be documented.

The software requirements specification serves as a central project document - a defining element with relationships to other elements of the project plan. The requirements include both technical (the behavior of the application) and non-technical (other project requirements including schedule, budget, etc.) requirements. In addition, acceptance criteria, which are the tests and measures that will be used to validate that the software meets its requirements, must be established and documented.

Software requirements must be controlled to establish a baseline for engineering and management use.

In order to accomplish these objectives, adequate resources and funding must be provided for managing requirements. Members of the software engineering group and other affected groups should be trained to perform their requirements management activities. Training should cover methods and standards as well as training activities designed to create an understanding on the part of the engineering team as to the unique nature and problems of the application domain.

Requirements are managed and controlled and serve as the basis for software plans, work products, and activities. Changes to the requirements are reviewed and incorporated into the project plans. The impact of change must be assessed and negotiated with the affected groups. In order to provide feedback on the results of these activities, and in order to verify compliance, the CMM provides guidelines for measurements and analysis as well as activities for verifying implementation. Suggested measures include:

Members of the team must be trained to perform their requirements management activities.

1. Status of each of the allocated requirements
2. Change activity of the requirements, cumulative number of changes
3. Total number of changes which are open, proposed, approved, and incorporated into the baseline.

Measures should be established including status of each requirement.

Continuous Improvement - Requirements Management Under CMM Level III

One of the most enlightened aspects of the CMM is its understanding that the process of requirements management is not simply a-document-it-up-front-and-go waterfall model. With the CMM, requirements are living entities at the very center of the application development process. Not surprisingly, the process of effective requirements management appears at virtually all levels of the process model and within many key process areas. As an organization moves to Level 3, the focus is on managing software activities based on a defined and documented standard practice. Key process areas for Level III include Organization Process Focus, Organization Process Definition, Training Program, Integrated Software Management, Software Product Engineering, Intergroup Coordination, and Peer Reviews. The Software Product Engineering Key Practice is designed to cause an organization to integrate all software engineering activities to produce high quality software products effectively and efficiently. The Software Engineering Key Practice states that

> ***" The software requirements are developed, maintained, documented, and verified by systematically analyzing the requirements according to the projects defined software process."***

The analysis process is necessary to ensure that the requirements make sense, that they are clearly stated, complete and unambiguous, consistent with each other, and testable. Various analysis techniques are suggested including simulations, modeling, scenario generation, and functional and object-oriented decomposition. The results of this process will be a better understanding of the requirements of the application, which are then reflected in revised requirements documentation. In addition, the group responsible for system and acceptance testing also analyzes the requirements to ensure testability.

Requirements must be analyzed to determine completeness, consistency and testability.

The resulting software requirements document is reviewed and approved by the affected parties to make sure that the points of view represented by these parties is included in the requirements. Reviewers include customers and end users, project management and software test personnel.

In order to manage change in a controlled way, the CMM also calls for placing the software requirements document under configuration management control.

The software requirements document should be placed under configuration management.

Requirements Traceability

> ***"Consistency is maintained across software work products, including the software plans, process descriptions, allocated requirements, software design, code, test plans, and test procedures."***

Under the CMM, all worthwhile software work products are documented, and the documentation must be maintained and readily available. The software requirements, design, code, and test cases are *traced* to the source from which they were derived and to the products of the subsequent engineering activity. Requirements Traceability provides a means of analyzing impact before a change is made, as well as a way to determine what components are affected upon processing of a change. Traceability also provides the mechanism whereby the adequacy of test coverage can be readily determined.

Software requirements, design, code, and test cases are traced from the source and to the products of the subsequent activity.

All approved changes are tracked to completion. The documentation tracing the allocated requirements is also managed and controlled. Measurements are made to determine the functionality and quality of the software products and to determine the status of the software activity. Example measurements include:

1. Status of each allocated requirement throughout the lifecycle
2. Change activity of the allocated requirements
3. Allocated requirements summarized by category

Managing Change

The CMM recognizes that change is an integral part of software activity. Indeed, we all now understand that the concept of "freezing the spec" is as old and obsolete as the generation of computers to which we tried to apply this philosophy. In place of frozen specifications, we now strive for a stable baseline of requirements which are well elicited, documented, and placed into systems which provide support for managing change. Specifically, the CMM calls for:

1. As understanding of the software improves, changes to the software work products and activities are proposed, analyzed and incorporated as appropriate.
2. Where changes to the requirements are needed, they are approved and incorporated before any work products or activities are changed.
3. The project determines the impact of change before the change is made.
4. Changes are negotiated and communicated to the effected groups.
5. All changes are tracked to completion.

The project determines the impact of change <u>before</u> the change is made.

All changes are tracked to completion.

Summary

The CMM provides a comprehensive view of the activities which must be applied to improve software quality and increase productivity. Requirements Management is an integral part of this process wherein requirements serve as living entities which are at the center of development activity. Once elicited, requirements are documented and managed with the same degree of care that we provide to our code work products. This process puts the team in control of their project and helps them manage both the project and its scope. Lastly, actively managing changing requirements keeps the project under control and helps assure the reliable, repeatable production of high quality software products.

Bibliography

[1] Paulk, M., et al, “Capability Maturity Model , Version 1.1”, *IEEE Software, 10, 4* (July 1993), pp. 18-27.

[2] Humphrey, W.S., *Managing the Software Process*, Reading Mass,Addison Wesley, 1989

[3] Paulk, M., et al, “Capability Maturity Model for Software, Version 1.1,” Software Engineering Institute, Pittsburgh, PA, SEI-93-TR-024

[4] Humphrey, W., et al., “Software Process Improvement at Hughes Aircraft,” *IEEE Software, 8*, 4 (July 1991), pp. 11-23.

[5] Putnam, Lawrence, "The Economic Value of Moving Up the SEI Scale", Managing System Development, July 1994.