PREVIOUS NEXT

Appendix F. Requirements Management in the SEI-CMM and within ISO 9000:2000

PREVIOUS NEXT

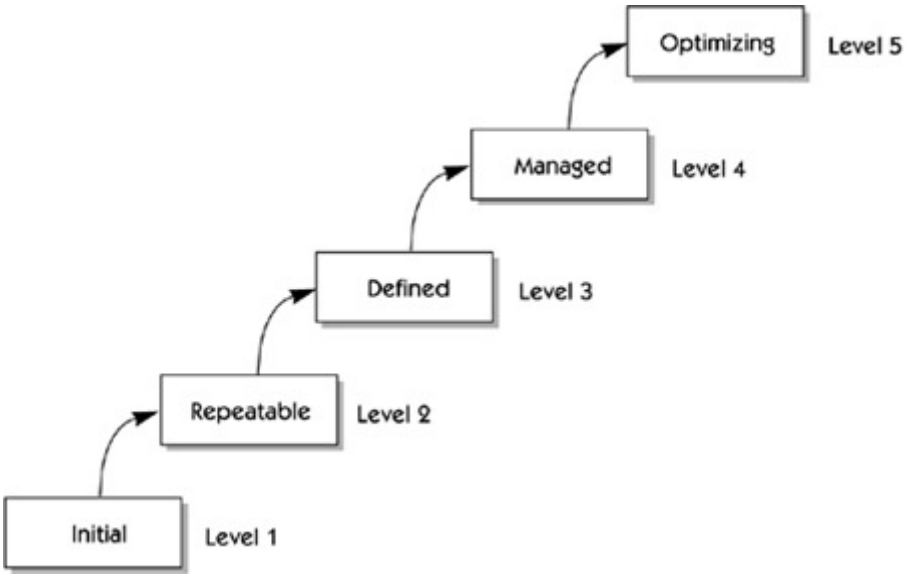
Team LiB

PREVIOUS NEXT

Requirements Management in the SEI-CMM

In November 1986, the Software Engineering Institute (SEI) at Carnegie-Mellon University began developing a process maturity framework to help developers improve their software process. In September 1987, the SEI released a brief description of the process maturity framework, later amplified in Humphrey's Managing the Software Process [1989]. By 1991, this framework had evolved into what has become known as version 1.0 of the Capability Maturity Model (CMM). In 1993, version 1.1 of the CMM was released [SEI 1993]. Version 1.1 defines five levels of software maturity for an organization and provides a framework for moving from one level to the next, as illustrated in Figure F-1. The CMM guides developers through activities designed to help an organization improve its software process, with the goal of achieving repeatability, controllability, and measurability.

Figure F-1. CMM maturity levels



Despite the ongoing debate and controversy about the advantages and disadvantages of the CMM, an accumulating body of data shows that adherence to the CMM and corresponding improvements in software quality have significantly lowered the cost of application development within many companies. By now, the CMM has been in use by many organizations long enough that meaningful and positive return-on-investment statistics are appearing. These payoffs should, ideally, provide results in productivity and significant reduction in time to market. In an era of increasingly competitive environments, any improvements to software productivity cannot be ignored.

The CMM provides a framework for process improvement that consists of "key process areas," or organizational activities that have been found, through experience, to be influential in various aspects of the development process and resultant software quality. Table F-1 identifies the key process areas of each of the five levels of the CMM. (The reason we're discussing all of this at length in this book

is that Table F-1 shows that the first key process area that must be addressed to move from level 1 to level 2 is requirements management.)

The CMM summarizes the process area of requirements management as follows: 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 development team and, as such, is the central document that 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 to be consistent with the requirements allocated to software. In this manner, the CMM moves the organization toward an integrated view wherein technical requirements must be kept consistent with project plans and activities. 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.

Table F-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 application functionality, cost, and schedule.	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

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 nontechnical (for example, schedule and budget) requirements.[1] 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.

[1] Note that the recommendation in this book excludes these project parameters from the application's requirements set.

In order to accomplish these objectives and to demonstrate compliance with the CMM process area of requirements management, 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.

The CMM further specifies that requirements should be managed and controlled and should serve as the basis for software plans, work products, and activities. Changes to the requirements should be reviewed and incorporated into the project plans, and 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

- Status of each of the allocated requirements
- Change activity of the requirements, cumulative number of changes
- Total number of changes that are open, proposed, approved, and incorporated into the baseline

One of the most enlightened aspects of the CMM is its understanding that requirements management is not simply a "document-it-up-front-and-go" process of the sort often prescribed in the waterfall methodologies of the 1970s (Chapter 3). With the CMM, requirements are living entities at the 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 on the CMM scale, the focus is on managing

software activities based on defined and documented standard practices. Key process areas for level 3 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 project's defined software process" [SEI 1993].

The analysis process is necessary to ensure that the requirements make sense and are clearly stated, complete and unambiguous, consistent with one an 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.

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 are 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.

Another important concept in the CMM is traceability. 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 when processing a change. Traceability also provides the mechanism for determining the adequacy of test coverage.

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

- Status of each allocated requirement throughout the lifecycle
- Change activity of the allocated requirements
- Allocated requirements summarized by category

Finally, the CMM recognizes that change is an integral part of software activity in any development project. In place of frozen specifications, we instead strive for a stable baseline of requirements that are well elicited, documented, and placed into systems that provide support for managing change. Specifically, the CMM requires the following.

- As understanding of the software improves, changes to the software work products and activities are proposed, analyzed, and incorporated as appropriate. Where changes to the requirements are needed, they are approved and incorporated before any work products or activities are changed.
- The project determines the impact of change before the change is made.

- Changes are negotiated and communicated to the affected groups.
- All changes are tracked to completion.

In summary, the CMM provides a comprehensive view of the activities that must be applied to improve software quality and to increase productivity. Requirements management is an integral part of this process, wherein requirements serve as living entities that 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 its project and helps team members manage both the project and its scope. Lastly, actively managing changing requirements keeps the project under control and helps ensure the reliable, repeatable production of high-quality software products.

Although all of this provides an important "validation" of the concept of requirements management, along with some high-level advice for inserting requirements-oriented processes into the development lifecycle, it doesn't tell us how to do requirements management. The detailed activities of eliciting, organizing, documenting, and managing requirements are the subject of this book, and these activities have been influenced by the CMM framework.

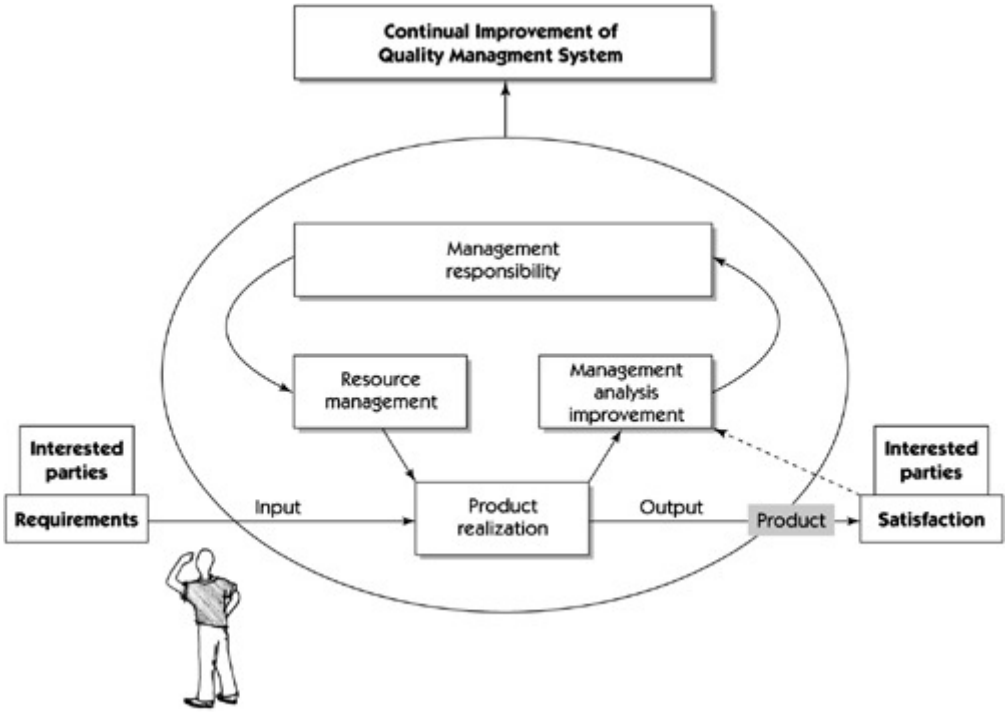
Team LiB PREVIOUS NEXT

Requirements Management in ISO 9000:2000

For the past two decades or so, a number of organizations around the world have been applying a series of comprehensive quality management standards known as ISO 9000 to improve operating efficiency and productivity and to reduce costs. In December 2000, these documents were revised to conform more closely to the lessons learned in the previous five years, and the resulting update to the standards became known as ISO 9000:2000. As opposed to the CMM, which addresses software development exclusively, the ISO standards are a much broader set of standards intended to cover virtually all business process activities in all businesses that engage in domestic and international trade of any sort. As such, ISO standards apply equally well to a small import company that simply resells products manufactured by others and to the largest global manufacturers of goods and services. In so doing, ISO has tremendous breadth and must be "all things to all people," but it cannot possibly contain the depth of coverage for any specific area as does a standard, such as the CMM, which is designed for a specific business process.

The standards apply to all facets of operations, from sales order activity to customer support, as well as product development activity and the role of software suppliers (developers) in the process of producing goods and services that depend on software. In this respect, the ISO 9000:2000 revisions are particularly significant to this book because the changes had the effect of shifting ISO 9000 into much more of a process orientation and away from its audit orientation. Figure F-2 illustrates how the focus has shifted to the implementation of a process that has continual feedback loops and therefore provides a built-in mechanism for continual quality improvement. And when the process being analyzed is product development, the input is based on requirements derived from the "interested parties," that is, stakeholders, to help ensure that the resultant solution meets their needs. So yes, an effective requirements management process stands at the very front of this business process, and in that way the philosophy of ISO is consistent with the philosophy of this book.

Figure F-2. ISO model of the process approach (adapted from ISO 9000:2000)



Today, ISO 9000 has been adopted by the European Community as EN29000 and has become an important factor for international trade; organizations wishing to do business in Europe, for example, often have to demonstrate ISO 9000 certification. Thus, in today's global economy, many American businesses have adopted ISO 9000 to ensure their credibility in the world marketplace. Certification to ISO standards requires an on-site assessment by an independent, ISO-approved assessor, so adopting such a set of standards represents a significant commitment. Companies are reassessed periodically to maintain their certification. ISO 9000 consists of four primary quality standards:

1. ISO 9000:2000 Quality management systems: Fundamentals and vocabulary (supersedes ISO 8402 and ISO 9000-1)

2. ISO 9001:2000 Quality management systems: Requirements (supersedes ISO 9001:1994, ISO 9002, and ISO 9003)

3. ISO 9004:2000 Quality management systems: Guidelines for performance improvement (supersedes ISO 9004-1)

4. ISO 19011: Guidelines on auditing quality and environmental management systems (supersedes parts 1–3 of ISO 10011, ISO 14010, ISO 14011, and ISO 14012)

Within these documents, ISO 9001 contains a number of specifications that focus on using an effective requirements management process to control and improve the quality of a product or service. In addition, the document now makes a distinction between product-oriented requirements and process-oriented requirements. Again, this philosophy is consistent with the definitions we provided for quality in Chapter 29, wherein we described software project quality as the characteristic of having demonstrated the achievement of producing a product that meets or exceeds agreed-on requirements (as measured by agreed-on measures and criteria) and that is produced by an agreed-on process.

The same document also stipulates that the information thus provided to the supplier (which we've described as the "developer" throughout this book) should include all performance, safety, reliability, security, and privacy requirements (a subset of the nonfunctional requirements described in this book) that collectively determine whether the delivered system is acceptable.

Like the CMM, ISO 9000 standards have been the subject of considerable debate, particularly in U.S. organizations that worry about the possibility of the standards degenerating into a bureaucratic demand for excessive documentation. Our purpose here is not to endorse or attack ISO 9000; like all such commonsense concepts, it can be used or misused. But to the extent that many organizations are adopting ISO 9000 because they think it's a good idea or because it's a necessary prerequisite for doing business in Europe and other parts of the world, it's interesting to note the emphasis that the standard puts on requirements management. For example, ISO 9000 emphasizes the need for mutual cooperation between the customer and the developer for software systems. Specifically, it calls for:

- Assignment of people from both groups to be responsible for establishing requirements
- Establishment of methods and procedures for agreement and approval of changes to the requirements
- Efforts to prevent misunderstandings of the requirements
- Establishment of procedures for recording and reviewing the results of discussions about the requirements

Although it's easy to dismiss all of this as obvious and commonsense, remember what happens during the assessment required to achieve certification. An assessor will visit the organization and ask, "Where are your methods and procedures for approving changes to the requirements? Show them to me in writing. Let me visit some project teams and make some spot-checks to ensure that the procedures are actually being followed."

ISO 9000 also stipulates that the input to the development phase of a project—the lifecycle activity in which technical design and programming usually take place—should be defined and documented. These "inputs" are, of course, requirements, and ISO 9000 also states that the requirements should be defined so that their achievement can be verified. The use-case-to-test-case approach we have described could be of real value here. ISO 9000 also calls for processes to ensure that incomplete, ambiguous, or conflicting requirements will be resolved in an orderly fashion. Your team may wish to apply an iterative and incremental process for successive refinement of requirements as we have described.

In summary, the ISO 9000 emphasis on requirements at the beginning of a development effort is intended to help ensure that, if the technical design and development efforts are carried out in a disciplined fashion, the organization is more likely to produce a system that meets specifications, or requirements, rather than relying on frantic testing and validation activities at the end of the lifecycle to assure quality. With respect to software development at least, that also nicely summarizes the purpose of this book!

Like the SEI-CMM, ISO 9000 doesn't tell you specifically how to actually do requirements management. Rather, your team members will be obligated to define, implement, and adhere to an effective requirements management process that they themselves declare to be sufficient for the intended purpose. But armed with the procedures and techniques described in this book, and with Chapters 30 and 31 as a comprehensive summary and process guide, your team will be able to create a comprehensive requirements management approach that should satisfy the most demanding ISO 9000 or CMM assessors.

PREVIOUS NEXT