Implementing the Capability Maturity Model for Software Development

Continuous support for a software development improvement effort requires at least two things: a clearly defined improvement model and success at applying the model in the organization. One HP division was able to apply one such model and achieve measurable success on several product releases.

by Douglas E. Lowe and Guy M. Cox

Manufacturing entities are always looking for more efficient ways of producing products because they realize that an efficient process yields lower costs, better quality, and increased customer satisfaction. Software manufacturers are no different from their hardware counterparts in that they want to use the best software development process available. The Capability Maturity Model (CMM) for software, developed at the Software Engineering Institute (SEI) at Carnegie-Mellon University,* is a process model that provides excellent guidance to improve software development processes.

The Model

CMM is used to evaluate and improve the way software is built and maintained. First released in 1987, CMM was originally based on the experience of members of SEI. CMM has been continuously improved and refined since 1987 through successive revisions based on industrywide and worldwide input. Yet, even though it is based on experience, it is only a model, which is an abstract, general framework describing the processes used to develop and maintain software. Like any model, it requires interpretation to be used in a specific context. The approach used by CMM is to describe the principles and leave their implementation up to the managers and technical staff of each organization, who will tailor CMM according to the culture and the experiences of their own environment.

Perhaps the most well-known aspect of the CMM is its description of five stages, or maturity levels, of an organization's software process (see Fig. 1). The first level of the software development process, referred to simply as the *initial level*, is described as ad hoc, poorly controlled, and often with unpredictable results in terms of schedule, effort, and quality. At level 2, the *repeatable level*, the outputs of the process are consistent (in terms of schedule, effort, and quality) and basic controls are in place, but the processes that produce those results are not defined or understood.

Level 3, the *defined level*, is the point at which the software engineering practices that lead to consistent output are known and understood and are used across the whole organization. The *managed level*, or level 4, is where the defined elements from level 3 are quantitatively instrumented so that level 5, the *optimizing level*, can be achieved. Level 5 exists in organizations in which the development process operates smoothly as a matter of routine and continuous process improvement is conducted on the defined and quantified processes established in the previous levels.

Thus, CMM is seen as a maturity or growth model in which an organization works its way up the five levels and, even after having attained level 5, is still in the process of continually improving and maturing.

Each of the five levels is also defined by the key processes associated with it. There are 18 key process areas that make up the five levels (see Fig. 1). These processes were chosen because of their effectiveness in improving an organization's software process capability. They are considered to be requirements for achieving a maturity level.

Managerial processes are those that primarily affect the way management operates to make decisions and control the project. Technical processes are those that primarily affect the way engineers operate to perform the technical and engineering work.

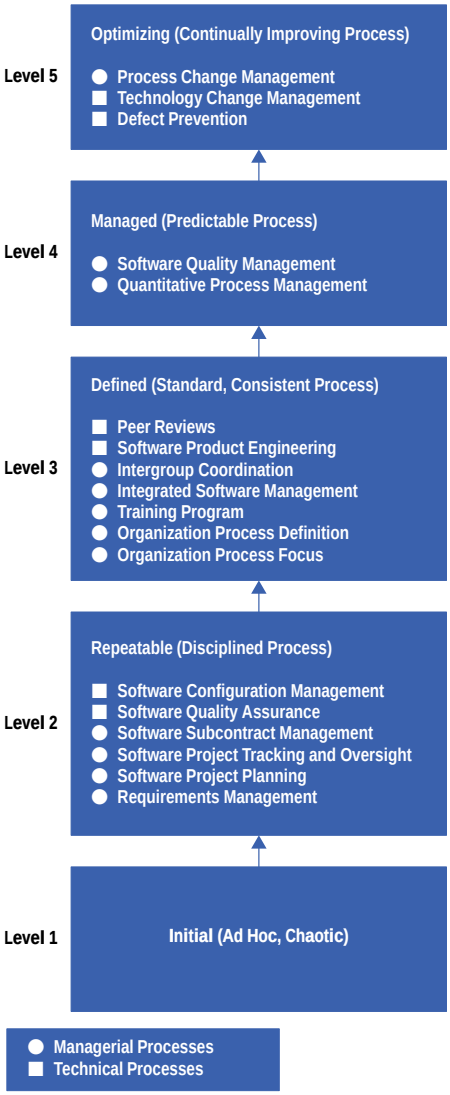
CMM provides a structure for each of the key process areas (see Fig. 2). Also, each key process area has one or more goals that are considered important for enhancing process capability. Fig. 3 shows the goals for three of the key process areas in level 2.

Finally, each key area has five common features or attributes:

- *Commitment to perform* describes the actions needed to ensure that the process is established and will endure and typically involves policies and senior management sponsorship.
- *Ability to perform* describes the preconditions that must exist in the project or organization to implement the software process competently. Ability to perform typically involves resources, organizational structures, and training.

* SEI is sponsored by the U.S. Department of Defense and was established in 1984 by the U.S. Congress as a federally funded research organization. Its mission is to provide leadership in advancing the state of the practice of software engineering to improve the quality of systems that depend on software.

Fig. 1. *The five levels of the Capability Maturity Model (CMM). The items listed at each level are called key process areas. These areas determine an organization's software development maturity.*



- *Activities performed* describes the roles and procedures necessary to implement a key process area. These typically involve establishing plans and procedures, performing the work, tracking it, and taking corrective actions as necessary.
- *Measurement and analysis* describes the need to measure the process and analyze the measurements.
- *Verifying implementation* describes the steps needed to ensure that the activities are performed in compliance with the process that has been established. Verification typically encompasses reviews and audits by management and software quality assurance.

The intent of CMM is to describe what needs to be done to develop and maintain software reliably and well, not how to do it. CMM further describes practices that contribute to satisfying the intent of these attributes for each key process area. Any organization can use alternative practices to accomplish the CMM goals.

The Challenge

It usually takes most organizations about two to three years to go from level-1 to level-2 CMM compliance. However, based on very sound business reasons, our general manager at HP's Software Engineering Systems Division (SESD) committed us to reaching level 3 in 36 months. To show a commitment to this aggressively scheduled task, three people were assigned to the project.

During the planning stage, we discovered that because this was such a reasonable program, we could complete level 2 in less than 12 months with less than three full-time people. Perhaps more important, we found that this program could provide immediate benefits.

Fig. 2. *The elements that make up the structure of the level-2 key process area: Software Project Planning. Each key process area has a similar set of elements.*

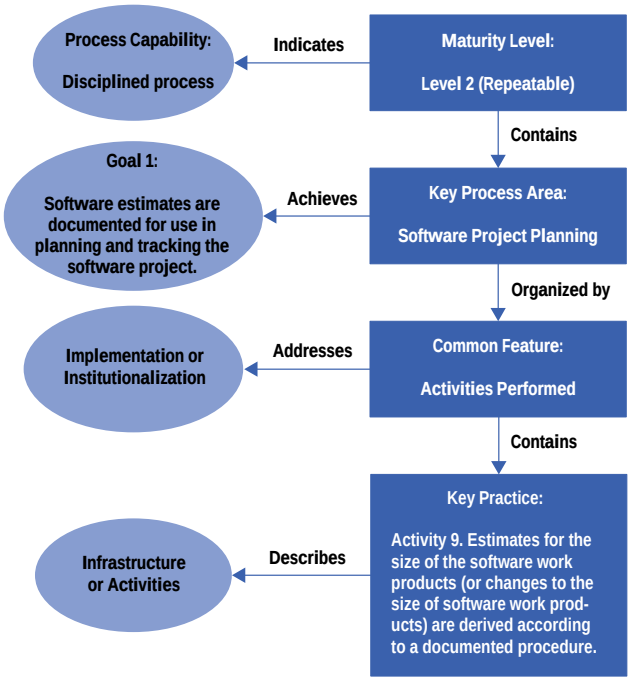


Fig. 3. *The list of goals for three of the key process areas for level-2 CMM compliance.*

Level 2: Repeatable. Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

Requirements Management

1. Requirements are controlled to establish a baseline for engineering and management use.
2. Plans, products, and activities are kept consistent with the requirements.

Software Project Planning

1. Estimates are documented for use in planning and tracking.
2. Project activities and commitments are planned and documented.
3. Affected groups and individuals agree to their commitments related to the project.

Software Project Tracking

1. Actual results and performances are tracked against the software plans.
2. Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the plans.
3. Changes to software commitments are agreed to by the affected groups and individuals.

In this article we describe how our product teams reached level-2 CMM compliance within a few months of starting the project, beginning with investigating the project in September and continuing with implementation in November 1994. By May of 1995 we had completed two deployment cycles with the product teams, and our internal audits of the teams' processes verified that we were operating at level-2 CMM. After several more audits of all the product teams, we found that the organization was operating at 100% level-2 CCM by August 1995. By May 1996 SESD had taken major steps toward achieving level 3 and is on track to achieve level 3 in 36 months. We expect that our description of how we accomplished this will provide insight for other organizations trying to achieve similar improvement in their software development processes.

The Improvement Project

HP's Software Engineering Systems Division (SESD) produces UNIX® software development tools, including SoftBench, C, C++, COBOL, UIM/X, HP Distributed Smalltalk,[1] and configuration management tools. At SESD, software engineers work

together in cross-functional teams of 10 to 15 engineers. The cross-functional teams are made up of representatives from R&D, marketing, and learning products. These teams report to two business teams, one for the technical market and one for the commercial market. A combined quality and productivity function completes the organizational picture.

For several years SESD has attempted to change and improve its software engineering process. However, like most organizations the development priorities were to get products out first and work on improvement if time permitted. Usually, there was very little time to work on improvement. The development priority for SESD is still to get the product out (as it should be), but the software engineering process is seen as an integral part of achieving that priority.

When we began this project SESD had in place and in use a standard software life cycle, excellent software development tools, and good practices relative to configuration management, defect management, inspections, coding, and testing. SESD also had an excellent customer satisfaction program in progress, and a basic metrics collection program in place.

However, there were still weaknesses in our engineering process. It took a long time to define products and our design process needed some improvement. The life cycle was defined, but there was a lack of procedures for performing work and a lack of engineering discipline for following defined processes. As a result, products were consistently taking longer than expected, product completion dates were missed, and many features appeared in a product that weren't originally planned.

There was expert help available from HP Corporate Engineering's software initiative program. HP's software initiative program has built expertise in software process areas that are critical to software improvement, and we were able to enlist the help of this organization in the early stages of our project.

HP's software initiative group is a team of engineering and management experts who deliver knowledge and expertise through consulting in software engineering practices. The software initiative team works with multiple levels of the organization to optimize the organization's investment in development capability, accelerating the rate of making lasting strategic improvements and reducing the risk of change.

Beginning with the End in Mind

As mentioned above, we knew that it would be difficult to achieve level-3 CMM compliance in 36 months based on the experiences of other organizations inside and outside HP. Our problem was one of finding a way to institute process changes in an orderly way without adding major risks to the product teams' execution of their projects.

Adding to the sense of urgency were the very real business goals that needed to be achieved for SESD to be fully successful. The critical business issues were product time to market and quality improvement. These issues were evident in the past performance of the product teams in delivering products within an 18-to-24-month window and in the difficulty of delivering products that addressed major customer satisfaction issues. Responding to these critical business issues provided the real endpoint and goal that the entire organization could work to achieve.

The SoftBench[2] product team,which was the first group within our division to begin applying CMM level 2, had a business goal of releasing an update in a 12-month cycle, which would mean a 6-to-12-month reduction in typical cycle time. The team also had additional goals of reducing the number of lines of code in the product, delivering three major customer satisfaction enhancements and three competitive enhancements, and fixing all major customer reported problems.

We designed the software improvement project to help support the goals of the SoftBench product. The life cycle and the processes were defined to map into the objectives of a 12-month release cycle, provide methods for requirements analysis and specification in a short period, and provide aggressive management of defects during the development process.

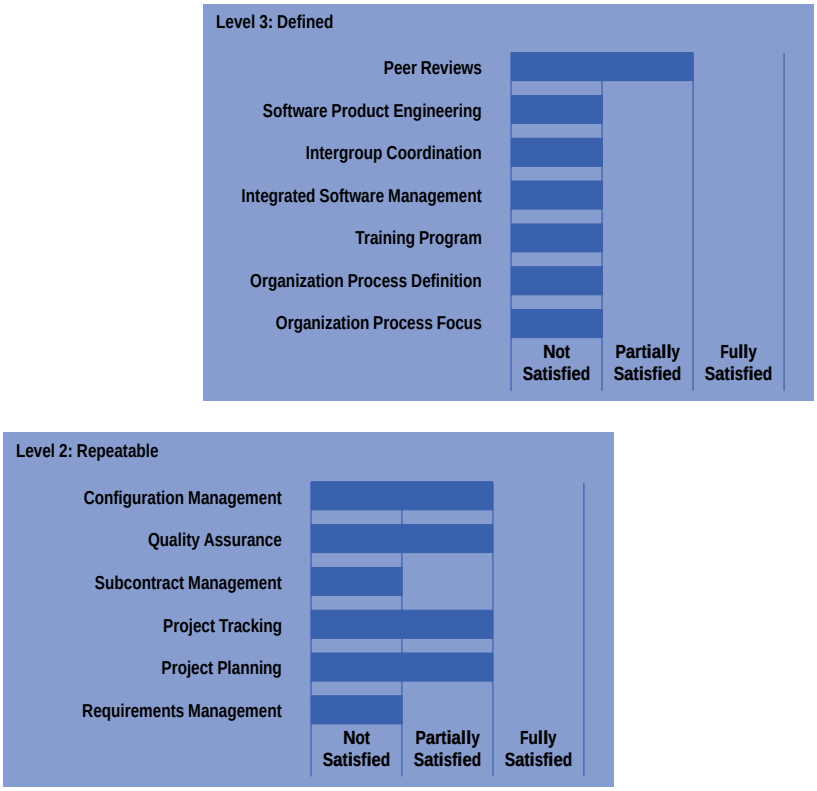
Investigation: Training and Planning

To evaluate our current practices we used a technique called a *software process profile,* which was developed by HP in collaboration with the Software Engineering Institute at Carnegie-Mellon University. The process profile is an assessment of the state of an organization's software development process, identifying strengths and weaknesses, highlighting the process improvements the organization values most, and recommending areas for change. The profile uses CMM as the standard for evaluating the software process. Using questionnaires and open-ended interviews, it provides results for all eighteen of the key process areas shown in Fig 1.

To complete the profile, engineers and managers in an organization must fill in a questionnaire that is designed to evaluate that organization's software process maturity against the CMM requirements. The results of this questionnaire are compiled into a software process profile for the organization. Fig. 4 shows SESD's software process profile.

We selected a representative group of 30 engineers and managers to participate in the assessment and, over a period of two days, we provided a short overview training session on CMM and a two-hour period for small groups to answer the assessment questions.

Fig. 4. *A software process profile indicating the compliance of SESD to the requirements of levels 2 and 3 of CMM at the time the software improvement project began.*



An example of a typical question from the project tracking and oversight area asks the participant:

"Does someone review the actual project results regularly with your section and lab managers?"

1–Almost Always 2–Often 3–Seldom 4–Almost Never 5–Doesn't Apply 6–Don't Know

For SESD, the process profile results (Fig. 4) indicated that four level-2 process areas were partially satisfied and two areas, requirements management and subcontract management, were areas that were not satisfied at all. Out of the seven level-3 processes, SESD partially satisfied only one area—peer reviews. These results meant that our organization would need to focus on implementing improved practices for all of the level-2 key process areas.

After the results were processed, we held several review sessions with engineers and managers to explain what the results meant. In one of the early sessions we asked the attendees to help us identify the benefits of improving our performance in each area and the roadblocks to achieving the requirements in each area. For example, Table I shows the assessment results from the software project tracking and oversight key process area.

We found that the assessment process was an important element for describing the improvements needed and determining how we should go about making the improvements. It quickly focused the entire organization on an improvement goal for our software processes and provided the starting point for getting participation in planning what actions we needed to take.

Applying the CMM Model Effectively

One of the critical steps in getting started was to understand CMM in detail. Over a period of several weeks three process consultants met daily to review the CMM specifications, develop our interpretation of the model, and translate it into language that could be applied within our organization.* This was an important step because CMM contains many practices that are better suited to large organizations and it was necessary to interpret which of these practices would apply to our organization. Also, during these meetings ideas of how best to deploy these practices were developed.

The process consultants examined each of the key process areas and decided what would be required to define our processes in a way that satisfied the requirements for level-2 CMM compliance. Several discoveries were made in the course of this work:

- SESD already had many process assets that could be leveraged to support the key process area requirements. These assets were our software life cycle, our formal documentation for patches, releases, and defect management procedures, and several informal methods of documentation for specifications, design, and testing.

- We discovered the need for a *process architecture* when we attempted to describe what deliverables would be needed to support the definition of our processes. A process architecture is analogous to other types of engineering architectures (e. g., buildings, software, hardware, etc.). It describes the layout of components needed to build a system for a specific purpose. Fig. 5 shows the process architecture elements we used to create parts or all of the documentation needed to provide SESD specifications for creating work products and performing software development activities.
- Level-2 CMM permits each project team to document the procedures that will be used in developing products. By writing a general procedure to cover the way product development is generally performed and then customizing the procedure for each project team, if necessary, we realized that we could save several months of effort and speed up the eventual move to level-3 CMM, where organizational processes must be standardized.

Table I
Assessment Results for Software Project Tracking and Oversight

Survey Questions	Percent of Survey Respondents		
	Fully	Partial	Not
Are the project's planned activities and deliverables tracked (e.g., schedule, effort, and tests)?	43	24	33
Are the actual results compared to your estimates throughout the project?	10	19	71
If there is a variance, does someone take corrective action or explain why the variance has occurred?	14	38	48
Are changes to the activities, deliverables, and schedule discussed with the people who will be affected?	19	58	23
Does someone review the project results regularly with your section and lab managers?	18	18	64

Formal Project Planning and Decisions

To give this improvement project the greatest chance of success and reduce the overall risk for the project, we developed a formal project plan covering every phase of the definitional work and the timing for deployment of the processes. This plan was reviewed and approved by the division staff before beginning the implementation.

Several key decisions needed to be made about how the project deliverables would be designed, reviewed, approved, and deployed into the product development team's operations.

Several models for process creation, approval, and deployment were examined and discussed before it was decided to use a define-deploy approach that mapped into each development project's life cycle.** This allowed the improvement project team to stage the work by life cycle phase (i.e., requirements, design, implementation, and test). This model also provided a structure within which process deliverables could be refined and improved.

Measuring Progress

Methods for measuring the progress of the project needed to be established. The only way to do this was through auditing the product development teams after each major checkpoint. Thus, we decided that a thorough audit of how the product development teams conducted work should be done after checkpoints when the pressure to complete the checkpoint had subsided and the team would be more open to listening to the audit findings. These findings were reviewed with the management team and any major issues were addressed by assigning owners and developing action plans. The results of the audits were summarized and published within the division to let everyone understand the accomplishments of the program.

Measurement of progress in each key process area was performed by interviewing project team members and using a checklist of requirements for that phase of the project. This checklist was based on the practices needed to satisfy the goals for each of the key process areas that apply to level-2 CMM.

* The process consultants were originally members of the SESD software quality assurance group. They had extensive experience in software engineering practices management and software consulting.

** A define-deploy approach means that the processes are defined in the project team just before application. This allows processes needed for the requirements phase to be defined and deployed as the requirements phase of the project is executed.

Fig. 5. *The process architectural elements and their purpose.*

Element Type	Purpose
Policy	• Specifies what will happen • Sets the cultural expectations • "That's how we do things around here"
Procedure	• Specifies how it will happen • A set of steps for doing something • May specify who does what when
Checklist	• Specifies what or how in abbreviated format • A short form of procedures for easy reference or verification of actions
Template	• Specifies the content or quality of what will happen • Provides guidance for creating necessary work products
Training	• Provides organized information on processes that individuals need to perform their jobs • Covers policies, procedures, checklists, templates, or instructions • May be formal classroom or informal orientation
Work Product	• Specifies a plan or results • Created as an output of applying a defined process • May need to be "managed and controlled"

From previous experience, we realized that communication would play an increasingly important role for the project's success as new procedures and supporting documentation were developed. We decided that all of the documentation would be integrated into the software life cycle so that there would be just one place to find the information. Secondly, this documentation would all be online and the Mosaic browser coupled with our configuration management tool would be used to store and control the documents and provide an easily navigable interface for users.

Implementation: Managing the Change

Getting an organization to adopt the changes necessary for level-2 CMM compliance was an important step in implementing our new processes. Defining the new policies and procedures and then providing training in these new policies and procedures was necessary but not sufficient. Changing the processes involved changing the culture, and this is where it was critical to get everyone thinking about the changes in a positive way. We approached this aspect of change management by deciding we would try to accomplish the following goals:

- Demonstrate success with a phased approach
- Leverage existing processes and minimize some areas of change
- Make the contributions of everyone very visible.

Demonstrate Success with a Phased Approach

The software improvement project needed to show results as early as possible to capture the attention of the organization and to keep things focused on process improvement. To accomplish the objective of showing results early, the deployment stages of the improvement project were designed to be about three months each.* This tactic provided visible results and feedback to the organization by coinciding with the life cycle phases of the SoftBench product.

The implementation stage of the improvement project consisted of a series of short steps for defining, reviewing, and approving the policies, procedures, and templates before deploying them to the product development teams. Communicating what was expected, describing changes from the way we used to do things, and providing group or individual training in new methods were very important steps in the way we deployed the process changes. We knew that the project teams needed to understand the rules early so that the project could proceed smoothly.

An Example: The Requirements Phase

During the requirements phase of the product life cycle, the key process areas we worked on were practices for requirements management and project planning. By using readily available customer survey data, structured processes, management review milestones, and training, we were able to reduce the time for the requirements phase from what was historically a six-to-eight-month process to three months. For the SoftBench teams, it was important to gain a deeper understanding quickly of the new features demanded by our customers and to translate this information into work steps that would be needed in the design phase.

* Deployment was a term we used to mean communication, training, and consulting with the product development team, followed by application of the procedures or templates by the engineers and managers. Templates consisted of the structured outlines for the work products, such as design specifications, test plans, data sheets, and so on.

One of the problems the teams faced was reducing the list of possible things to accomplish to a few high-impact requirements that could be accomplished within the schedule. Thus, to collect requirements information, survey questionnaires based on a user-centered design methodology were created to facilitate rapid feedback from customers using telephone surveys. The process consultants designed standard templates for the engineers to describe the customer requirements and the Mosaic browser was used to post work in progress, allowing managers to review the work. Fig. 6 shows a portion of one of these templates.

Fig. 6. *A portion of a template containing a survey that the SoftBench product team used to solicit customer requirements.*

SoftBench 5.0 Written Survey

Environment:

1. What computer systems do you use for software development?

Make/Model/Memory _____ _____ _____

2. What types of operating systems do you use for software development and deployment?

Operating System	Software Development	Software Deployment
HP-UX	_____	_____
SunOS	_____	_____
Sun Solaris	_____	_____
PC Windows	_____	_____
Other:	_____	_____

3. What tools do you and others in your team use for software development? Where appropriate, please include a vendor name.

Requirements gathering
- Analysis _____
- Design _____

Implementation
- Understanding code _____
- Editing _____
- Compiling _____
- Debugging _____
- Testing _____

Configuration management _____
Project management _____
Release _____
Other key tools _____

4. What languages are used in your work group, in relative proportions? Please make answers in a column total 100%.

Language	Today	In 12 Months	In 24 Months
C	_____ %	_____ %	_____ %
C++	_____ %	_____ %	_____ %
COBOL	_____ %	_____ %	_____ %
Fortran	_____ %	_____ %	_____ %
Other	_____ %	_____ %	_____ %
_____	_____ %	_____ %	_____ %

5. Do you do have a mixed-language development? [ ] Yes [ ] No

If so, in what languages?

Your Tasks:

6. What type of software do you develop?

7. Do you develop programs, libraries, or both?

8. What are the major tasks you perform with regard to software development?

9. What tasks have you recently completed?

10. What tasks are you currently working on?

11. If you use SoftBench, what tasks do you use it for?

_____ Edit	_____ Static Analysis
_____ Compile/Build	_____ Code Generation
_____ Debug	_____ Configuration Management
_____ Performance Analysis	_____ Mail
_____ Comparing/Merging Files	_____ Tool Integration
	_____ Other

12. About how many lines of code are in each of the programs or libraries you develop? Indicate the number of programs and libraries that you have in each of the following categories.

Size	Programs	Libraries
<10K	_____	_____
11-50K	_____	_____
51-100K	_____	_____
101-300K	_____	_____
>300K	_____	_____

. . .

One criterion for determining what new features to include was an estimation of the resources and effort needed to design the new features. A standard procedure was provided for the engineers to do this (Fig. 7). These estimates were then available when the managers needed them for decision making and for the engineers to do more detailed planning of the design phase activities.

The decision making process was essential for narrowing the scope of the project and finalizing the work commitments for the next phase of the product development.

An Example: The Design Phase

During the design phase of the product life cycle, the key process areas were the practices in project tracking (i.e., managing and controlling work products, especially changes in requirements), and in project planning for the next phase. The success in applying the software life cycle and CMM level-2 processes to this phase of the work was evident from two major accomplishments. The first was that the team completed every aspect of the design work, including reviews and inspections, in three months. In the past, design specifications and design reviews were cut short because of schedule pressures. The team was also able to make decisions early in the project about eliminating features that would be too costly

Fig. 7. *A template for engineers to estimate the documentation delivery dates and code size of new software components.*

Tool/Component	External Specifications			Internal Specifications		Size Estimates	
	High Level	Prototype	Final	High Level	Low Level	Current	Estimated
A	mm/dd/yy	mm/dd/yy	mm/dd/yy	mm/dd/yy	mm/dd/yy	<n KNCSS>	<n KNCSS>
B	mm/dd/yy	mm/dd/yy	mm/dd/yy	mm/dd/yy	mm/dd/yy	<n KNCSS>	<n KNCSS>

n KNCSS = Number of KNCSS (Thousand Lines of Noncomment Source Statements)

to implement. For example, unit testing capability was a feature considered for the product, but it was eliminated during the requirements phase because of staffing trade-offs. This action substantially reduced the risk of schedule slippage later on.

Leverage Excellence to Minimize Change

As mentioned earlier, during the assessment and planning of the improvement project, we recognized that many practices used by SESD were already at level-2 CMM compliance. Therefore, it was important to leverage as many of the current practices and procedures to minimize the effort required and reduce the amount of change being introduced. We already had a standard software life cycle defined and in use, an excellent software development toolset, and good practices and tools in configuration management, defect management, inspections, coding, and testing. Part of our standard operations included a customer satisfaction survey and software metrics. These were all areas that were acknowledged as excellent and that should be maintained and supported.

The development environment consisted of a suite of tools integrated with SoftBench. UIM/X (user interface Motif) was used as a rapid prototyping tool. SoftBench provided the framework for editing, compiling, and debugging the code. Capabilities in SoftBench were also being used to assist in program understanding through call graphs and documentation of object-oriented designs for the new features. Integrated into SoftBench was a configuration management tool (SoftCM) for managing source code, and the ability to access our defect tracking tool DDTs from QualTrack. Having these tools already in place and used by the engineers was a major factor in maintaining developer productivity while making process changes in other areas.

Inspections and software metrics were already a well-established part of our culture. Although both of these areas are elements of level-3 CMM and we were working on level 2, we used them anyway because we did not want to lose the benefits we were achieving with these practices. Because inspections and metrics (defect tracking, schedule slippage, and effort reporting) were institutionalized in our engineering and project management practices, we recognized that this would be a distinct advantage for us when we went to level-3 CMM.

There were other opportunities to minimize change. These were in the areas of software configuration management, software quality assurance, and subcontract management. The first two areas needed only minor changes to be level-2 compliant. Our practices in these areas were robust enough but needed to be documented, with better definitions of roles and responsibilities. SESD wasn't doing any software subcontracting, so it was decided to leverage a best practice from within HP to document a starting point for future use.

Make Contributions Visible

Managing change requires good communication of the change, acknowledgment of the progress made, and encouragement toward the goal. After each product life cycle checkpoint, the software quality assurance team performed an audit by interviewing the product team members using a checklist for level-2 requirements. The software quality assurance members were actually the process consultants from SEI. They were able to bring experience and maturity to the audit interviewing, reporting, and follow-up consulting. Fig. 8 shows portions of the checklist for the requirements phase.

These audits had several major benefits. First, the project team knew ahead of time that a process they used during a phase would be objectively evaluated by an independent team. This had the effect of elevating the importance of using a defined process. Second, the audit interviews uncovered critical issues and risks for the product's development that were almost always a result of deviating from the project's plan or processes.

For example, during the audit we identified a problem with inadequate staffing for test planning activities. The project plan called for the completion of test plans before design was finished. The audit found incomplete test plans for the context feature changes before the design complete checkpoint. This introduced additional schedule risk because the schedule and staffing were not accurately estimated. It turned out that this part of the project was actually critical path during the implementation and testing phase.

These issues and risks were reviewed by the management team and decisions were made to take corrective actions. This had the effect of identifying tangible contributions of the level-2 model.

Fig. 8. *Portions of the audit checklist for the requirement phase. These checklists were used to assess the SESD life cycle against the level-2 CMM requirements.*

Checklist: SQA Project Audit Plan for Requirements Phase

Template

Requirements Management Checklist

- Responsibilities were assigned for developing and analyzing requirements.
- Staffing was sufficient for developing and analyzing requirements.
- Adequate training and tools were provided for developing and analyzing requirements.
- Requirements are documented in the project data sheet.
- Requirements were reviewed by affected individuals and groups.
- Issues related to requirements were reported and tracked.
- The project data sheet was reviewed and approved according to a documented procedure.

Project Planning Checklist

- Responsibilities were assigned for the planning requirements phase activities.
- Staffing was sufficient for planning activities.
- Adequate training and tools were provided for planning activities.
- Requirement phase activities were documented in a requirements phase plan.
- Estimates of schedule were prepared and included in the requirements phase schedule.
- A software life cycle was identified or documented in the requirements plan and the design phase plan.
- Work products for control of the project during the requirements phase were identified in the project planning documents.
- Commitments were negotiated with the business team managers and other affected groups.
- External commitments were reviewed and approved by the business team managers.
- Responsibilities were assigned for developing and maintaining the design phase plan.
- Adequate training and tools were provided for planning the design phase.
- A design phase plan was prepared according to a documented procedure.
- Design phase activities are documented in the design phase plan.
- Estimates of size, effort, and cost for design phase planning were prepared according to a documented procedure.
- Risks were identified in the design plan and contingency plans were developed.
- Requirements are traceable in the design plan.
- Issues related to design are reported and tracked.
- Design phase plans were updated to reflect changes in requirements.

Key Results and Benefits

The improvement project to achieve level-2 CMM has had several direct results. First, the cycle time for the SoftBench release was reduced from 18 to 24 months to 14 months. This resulted in a significant reduction in engineering time, and consequently, a saving in the cost of developing the product. While the 12-month release cycle required a little longer than planned, the commitment date at the time of design completion was met with no schedule slip and no reduction in product reliability standards. Across all of SESD's products there was a reduction from an average of 4.6 open serious defects at manufacturing release in 1994 to 1.6 open serious defects per product in 1995. In fact, the product team fixed all outstanding major customer service requests during this period. In addition, the product team reduced the overall code size by 12%, reduced documentation size by 35%, and delivered three major customer satisfaction improvements and three major competitive improvements. All of these are, of course, significant business results coming from the level-2 CMM process. On SoftBench alone there was a cost reduction of two million dollars per year, or a return on investment of approximately 9 to 1.

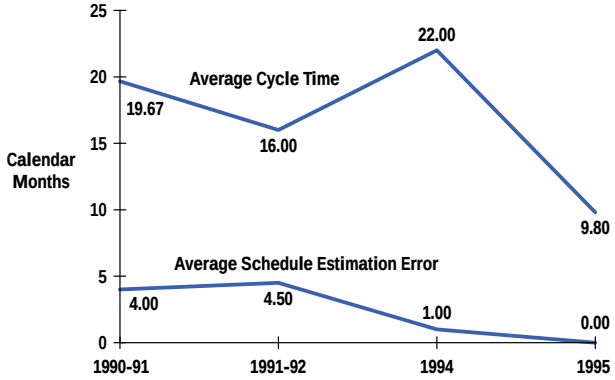
Other results of the improvement effort are that all of the product releases in 1995, which also met level-2 CMM compliance, were completed in under 12 months with no schedule slip. Fig. 9 shows the cycle times for similar projects over the past five years. The average cycle time for the 1995 projects was 9.8 months, which is a 46% reduction in cycle time from the running average of 18 months for previous years. The schedule estimation error (from design completion to project completion) was reduced to zero during 1995, compared with much higher errors in previous years.

Improved Execution. In past projects, the investigation phase usually lasted from 6 to 12 months, mainly because of the unstructured, exploratory nature of the process used. By adopting a formal structure (i.e., tasks and schedules), the investigation phase was reduced to four months.

Given the 12-month release cycle, it was critical to meet the intermediate phase deadlines to keep the project on track. In past projects, these milestone deadlines were consistently delayed by months. As a result of the careful planning and tracking processes used, the SoftBench team was able to meet the checkpoint deadlines within a very narrow margin of error (i.e., a few weeks).

Customer Orientation. Historically, our product requirements process had been focused on prototyping features and functionality that engineers identified through an infusion of "next bench" ideas. Product marketing would test these ideas with customers and use this feedback to select the best set of features to include in the next release. Everyone recognized the limitations of this approach in getting fresh ideas and direction for a product. Process improvement efforts were underway to define a user-centered design process that would really focus on what customers had asked for. The major change that occurred as a result of adopting the level-2 CMM practices is that we forced ourselves to adopt a new paradigm in the way we acquired and evaluated customer input.

Fig. 9. *The average cycle times (design complete to project complete) for projects similar to SoftBench over the last five years.*



The Ability to Respond to Changes. In most of our earlier projects, when changes in requirements or personnel occurred, there would be several weeks of confusion before revised plans or schedules could be started. In the new model of project management and level-2 practices, when a change occurs the management team knows that replanning must start immediately. Operating at level 2, the SoftBench team was able to estimate the impact of proposed changes and then schedule the time for engineers to replan and estimate the new schedule. When this occurred, we did not have the usual confusion about what to do. Instead, the team was able to respond with confidence and with very little lost time.

Conclusion

We believe that our circumstances and development environment are not unique. Many other organizations are trying various quality improvement programs for software development and finding it very difficult to make the changes necessary to be more rigorous and disciplined in their engineering practices. We also believe that we have discovered many of the essential ingredients to make a software improvement program succeed in a very short period of time.

Our work at SESD has convinced us that the Capability Maturity Model from SEI provides an excellent framework for defining software engineering process improvement for a small software organization. Achieving level-2 status has largely been a matter of establishing a direction with leadership from top management and then instituting a credible program for improving the practices used by software projects in planning and managing the work according to the CMM guidelines. We believe that other organizations can achieve similar results in one year if leadership and execution of the software improvement project are a priority for the management team.

References

1. E. Keremetsis and I. Fuller, "HP Distributed Smalltalk: A Tool for Distributed Applications," *Hewlett-Packard Journal*, Vol. 46, no. 2, April 1995, pp. 85-92.
2. *Hewlett-Packard Journal*, Vol. 41, no. 3, June 1990, pp. 36-68.

▶ Go to Article 2
▶ Go to Table of Contents
▶ Go to HP Journal Home Page