Applying CMM Project Planning Practices to Diverse Environments

Donna L. Johnson and Judith G. Brodman, *LOGOS International*

Many organizations today are using the Software Engineering Institute's Capability Maturity Model and experiencing difficulty applying it to their nontraditional development efforts. The authors show how to improve project planning processes using the CMM by focusing on its intent, instead of treating its practices as gospel.

Software developers often view themselves as creative individuals with a unique set of skills and methods they can apply to each new project. In keeping with this view, they frequently succumb to the not-invented-here syndrome and, as a result, end up redoing what someone has already solved with an allegedly newer and better approach. Introducing a structured software process improvement program based on the Software Engineering Institute's Capability Maturity Model[1] into this environment of self-proclaimed cowboys presents a challenge to even the most skilled champion. We often hear software developers say that their organization and applications are different, their projects have unique challenges, and their customers are more indecisive or inflexible than most. Because of these differences, they think the CMM does not apply to them.

Based on what we have seen, we do not dispute these perceptions. Not only do the individuals in an organization vary in personality, skill sets, experience, and methods of attacking problems, but the problem domain always has a unique twist to it—some anomaly that reinforces the saying that there is an exception to every rule. As a result, trying to apply the CMM (which the SEI patterned after the practices of a large aerospace company) equally to all organizations presents a formidable challenge. Nonetheless, we discovered in working with CMM-challenged organizations that, despite their apparent uniqueness, most of their projects tend to mirror one or more of the following characteristics: product-line development, maintenance, services, database development, desktop customization, small-project development, or schedule-driven development. Recognizing that an organization's projects might have one or more of these characteristics opens the door for that organization to use the CMM.

The project planning process in most organizations can satisfy the CMM no matter how diverse the process might appear to be from the model. In this article, we specifically target as examples projects in organizations with the characteristics from our above-described list. We also show how these organizations can focus on meeting the intent of the CMM's Software Project Planning Key Process Area (KPA) goals rather than on implementing each specific practice exactly as stated.

0740-7459/00/$10.00 © 2000 IEEE

Table 1

Common Characteristics of CMM-Challenged Projects

Characteristic	Description	Software project planning challenges
Product-line development	Company or organization formed around a product line. New development efforts build on initial product with enhancements or major product upgrades. Maintenance releases contain bug fixes and minor enhancements. Multiple customers and multiple versions of the product supported.	Determining the unit of work on which to focus planning efforts. Creating the recommended documentation with limited resources.
Maintenance	No new product development, except for occasional minor enhancements. Updated baselines released on a regular basis. Project resources and budgets allocated on level of effort basis over a fixed interval of time, usually one year.	Determining the unit of work on which to focus planning efforts. Satisfying CMM estimation goal with predetermined fixed resources and budgets.
Services	Organization provides services, on a contract basis, to other companies. Personnel trained in the organization's processes.	Satisfying CMM practices when the client process is in conflict with or substandard to the organization's process.
Database development	Delivered product is data. Any software developed in support of generating the data not delivered.	Tailoring CMM practices to apply to data generation.
Desktop customization	Target software developed by external organization or company. Organization's task is to customize software to its target environments. Customization ranges from setting preferences to developing interface software.	Tailoring CMM practices to apply to the customization of externally developed software (COTS products).
Small-project development	Common project size of one to six people; organization might be small as well. Staff assumes many roles to support the development effort. Management often has a technical role. Few, if any, overhead resources are available.	Implementing CMM practices with limited resources and for an organization and project structure that do not fit the practices.
Schedule-driven development	Schedule dates determined by external constraints, a group external to software, or management with insufficient staff input or insufficient requirements definition. Delivery dates often too aggressive for available resources.	Planning a project within the constraints of a fixed end date, which is often unrealistic.

Even though the CMM doesn't seem to apply to them, CMM-challenged organizations often have business reasons for using it, such as customer requirements, market competition, or a desire to take advantage of the benefits that using the model can reap. We intend to show that being different does not preclude an organization from applying the CMM to its development and planning practices. We selected project planning as our focus because it is the cornerstone of most of the CMM Level 2 KPAs, including Software Project Planning (SPP), Software Subcontract Management (SSM), Software Quality Assurance (SQA), and Software Configuration Management (SCM).

Project Characteristics

Before addressing how to apply the CMM to CMM-challenged organizations, we need to describe their projects and understand why they have so much trouble adopting the model. Unlike traditional software development projects, their projects might not perform all the life-cycle phases of specification, design, code, and test. Although traditional projects tend to deal with software development from cradle to grave, we see projects that deal with database generation, specification of desktop-software preferences, systems engineering studies, systems integration, and supply of services as their development activities. Also, some projects focus on maintenance—the end phase of the software life cycle. Coupled with these characteristics, we see projects that are small and do not have the resources that a large project does to perform all the CMM's activities, at least in the way the model suggests. We also see projects that are driven by schedules that are nearly impossible to meet. Table 1 describes these projects in more detail.

Project Definition

Although most people assume that the term *project* does not need defining, we discovered that the definition is misdirected in

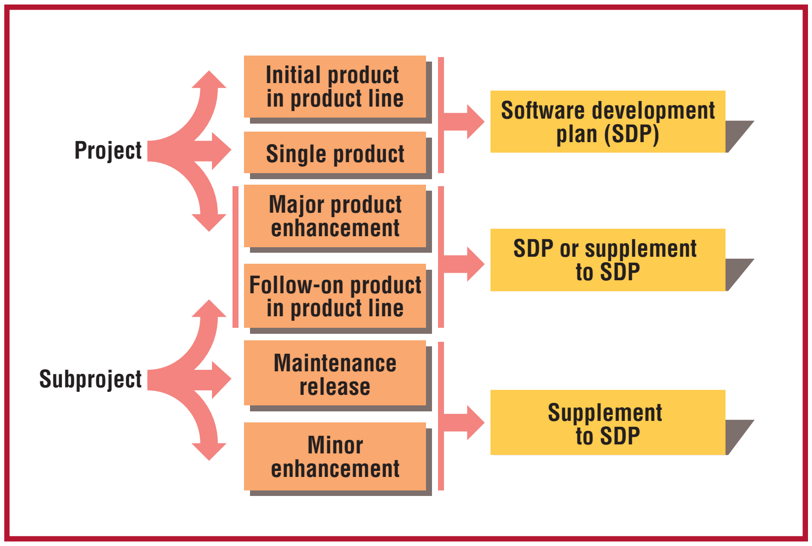
Project
Subproject
Initial product in product line
Single product
Major product enhancement
Follow-on product in product line
Maintenance release
Minor enhancement
Software development plan (SDP)
SDP or supplement to SDP
Supplement to SDP

Figure 1. A project definition.

many organizations. This misdirection is probably the most significant contributor to an organization's inability to apply the CMM to its work. How an organization defines a project can significantly impact the scope of activities it performs and the process artifacts it produces, because the model prescribes activities and artifacts for each unit of work that is carved out as a project.

With this in mind, we always ask an organization how many projects it currently has. We hear a wide range of answers, from one project to 85 projects to "I don't know." We were astounded to hear the number on the high end of the range, because the organization reporting the number had only 35 software developers. We realized that this particular organization was calling every product enhancement and bug fix a project. Considering that the CMM calls for a project plan and SQA and SCM plans for each project, it's no wonder that this organization felt overwhelmed in its attempts to use the CMM.

After discussions with this organization and a number of others, we redefined the concept of a project (see Figure 1). The development of a new product is a project, each subsequent product release is a subproject, and each enhancement and bug fix in the release is a task within the subproject. Major product enhancements, such as a version upgrade of an existing desktop software product or adding major product capability, are new projects if the product's scope changes significantly with the enhancement; otherwise, the major enhancement is still a subproject.

The project and subproject definitions work well for a maintenance organization or an organization that is built around a product line—cases where there are periodic product updates or releases. It also works well for any organization that produces multiple instantiations of the same type of product—for example, an organization that builds databases for different classes of transportation vehicles. The databases differ minimally within the same class but significantly between classes. According to our guidelines, an initial database development within any class is a project, but all subsequent databases within that same class are subprojects.

The greatest benefit of appropriately scoping a project becomes obvious when a project tries to satisfy CMM documentation guidelines. For example, the documentation it needs to support project planning is extensive. Each step of the planning process documents its results. We have seen organizations with 27-page templates for the project plan alone. The production and population of a document of this magnitude, along with a multitude of other documents (approximately 42 documents at Level 2 alone), is beyond the resources of many organizations and projects. The goal of these organizations should be to scale the quantity of documentation on projects to a level that is commensurate with their resources, their capability, and the size of their workload, yet still produce documentation that is useful to the organization and its projects.

One of the ways to accomplish this scaling is to develop only supplemental documentation (a supplement) for work that is similar in scope to previously documented work. Organizations should produce project documentation for the original product and major enhancements but generate only supplements to the original project documentation for subprojects. Little in the documentation changes from project to subproject, so generating supplements eliminates the overhead of duplicating identical information. We can apply this scaling to several of the project planning documents, such as the project plan and SQA and SCM plans. The initial plan contains the information we normally expect to see, whereas the supplement contains only information that changes for each instantiation of the project.

Maintenance projects are obvious candidates for supplemental documentation. They tend to vary little from year to year and can leverage a one-time maintenance plan (their equivalent of a project plan) developed for the project and then produce yearly maintenance plan supplements. Because the SQA and SCM efforts change very little as well, these projects can also document yearly SQA and SCM planning as supplements to project SQA and SCM plans, which organizations can further incorporate into the maintenance plan supplement to produce a single yearly planning document.

One organization we encountered had many small products in a maintenance mode and grouped several similar products together. The organization then created an umbrella project plan and a yearly supplement to cover all the projects in the group. Organizations can make similar project groupings for multiple small projects in an organization if they can establish commonalities between the projects. For example, one IT organization created project groupings of financial software, physical security software, and time-card accounting systems.

Project Estimation

The Software Project Planning KPA at Level 2 guides a project through estimating work, identifying and assessing risks, and documenting the results of the planning activities in a project plan. Planning activities in the CMM include estimating size, effort, costs, and computer resources and scheduling project activities based on these estimates. Projects develop estimates not only for the project's development and management activities but also for SQA and SCM at Level 2.

Cost Estimating

We see organizations allocate resources for a project on a level-of-effort (LOE) basis. Normally, from the CMM perspective, a project identifies required tasks and then estimates size, time, resources, and a budget based on these tasks. In LOE budgeting, however, the budget determines the workload. A maintenance project is not always able to control its workload over its project planning interval (usually a period of one year), so a fixed number of dollars are level-funded for that period. Defects are reported and enhancements requested during the year, and workload demands shift depending on the mix of priorities and the number of reported defects and enhancements.

Maintenance resources might be fixed, such as in a large organization with a dedicated maintenance group. In other cases, they might vary over the year depending on the number of high-priority tasks, with development resources from other projects temporarily reassigned to work on the high-priority items. Organizations schedule tasks such as bug fixing and enhancement implementation based on priorities. They often determine these priorities with the customer as *must have*, *good to have*, and *time permitting*. The estimated schedule for the planning interval normally falls somewhere in the middle to end of the good-to-have tasks. If the project is ahead of schedule, it implements some of the time-permitting tasks. If the project falls behind schedule, it implements fewer good-to-have tasks. Tasks that are not completed because of time and budget constraints become part of next year's planning exercise.

If a maintenance organization tries to apply the CMM estimating practices to its projects, it finds at first glance that the estimating practices do not fit. The CMM discusses estimating size based on historical data, estimating effort and cost from the estimated size and generating a schedule based on those estimates. Estimating in this way, however, is meaningless to a maintenance project:

- A size estimate for a maintenance project is the number of bug fixes and enhancements in a release instead of lines of code.
- Cost is an allocated dollar amount for the planning interval instead of an amount based on the workload.
- The effort is determined by what the budget allows instead of by workload demands.
- The schedule is a prioritized list of tasks instead of individual task milestones along a timeline.

Most maintenance organizations do not estimate cost and effort in the manner the CMM estimating practices suggest. The CMM Software Project Planning goal, however, does not prescribe a method for generating estimates—it requires merely that esti-

Normally, from the CMM perspective, a project identifies required tasks and then estimates size, time, resources, and a budget based on these tasks.

Size estimates are the cornerstone of all project estimates in the CMM.

mates be generated for use in planning and tracking the project and that those estimates be documented. Thus, to comply with the CMM, the maintenance project need only document, in the project plan and procedures, its alternative estimating practices as its estimating process, follow this process consistently, and then use the estimates it generates as the basis of its planning and tracking activities.

In other examples of LOE estimating, a maintenance project computes the SQA budget as a percentage (say 5%) of the project's development budget. The budget overrun one year was 2%, but the project has a historical record of meeting the 5% budget for SQA all other years. Another organization budgets a fixed yearly dollar amount across the organization to cover the SQA activity for all projects. If the organization has a higher-than-normal demand for SQA resources in a given year, some of the noncritical, low-priority projects might not receive SQA reviews. For the organization to be CMM compliant with this practice, it must write its SQA policy such that SQA reviews on noncritical, low-priority projects are optional—it conducts the reviews, time permitting, for those projects on a spot-check basis.

Another estimation practice we encountered is LOE budgeting for development activities across the whole organization. The organization is very small, with fewer than 10 software developers, and it focuses on allocating its resources across projects, not on how much those resources are costing for an individual project. The organization often shifts its resources across projects depending on project priorities and needs. Due to this shifting, estimating cost on a per project basis is meaningless because that cost will likely change, and tracking against a meaningless cost isn't productive. Dollars are thus allocated and tracked at the organization level as opposed to the individual project level, unless a project requires outside help because of schedule slips. The cost of the outside help is then charged to the project, and the project begins to track its cost overruns.

Based on the response from a group of over 20 assessors to this practice, which we included in an assessment case study in one of our training courses,[2] we found that this practice might be difficult to justify to an assessor. Most assessors indicated they look for cost tracking on the individual project, although some did reconsider after carefully considering the CMM's goals as opposed to its practices. If the organization documents this practice as its cost-estimating practice, if it follows what is documented, and if it successfully tracks and manages its budgets, the organization meets the intent of the CMM project planning estimating goal with respect to cost.

Size and Effort Estimating

Size estimates are the cornerstone of all project estimates in the CMM. The Software Project Planning KPA practices state that effort, cost, critical computer resources, and schedule are all related to the project's estimated size. However, few of the small projects we encounter even estimate size, and those that do estimate size in terms of number of screens, number of tables, or similar measures instead of the traditional lines of code, function points, or objects. These small projects usually estimate their effort based on past experience, and they determine that effort by computing a delta size to what they've done in the past, such as the increased number of screens or the product's increased functionality.

Obviously, this method won't work for all projects, particularly for medium- and large-size projects (where an analogy to past experience is not always a practical or accurate measure) or when the scope is significantly different. However, if small projects look beyond the CMM's practices to its goals, they can see that estimating effort using the analogy estimation technique is compatible with the CMM if they

- document their process of estimating effort based on analogy to previous projects,
- identify this process as the project's estimating process (or as one of the acceptable estimating practices in a standard process defined for the organization),
- state in their Software Project Planning policy that a project may waive size estimates in lieu of effort estimates based on analogy to previous similar projects, and
- document and use the generated estimates as the basis for managing the project.

We often hear that the CMM does not apply to commercial-off-the-shelf (COTS) projects. We see IT organizations, however, that are required to achieve Level 2 as part of a company-wide improvement initiative. Projects that customize COTS products for company use are at a loss as to how the CMM can apply to them, especially when that customization involves selecting and setting preferences for the products to run in the target environments. These projects do not develop software, so CMM practices in general and estimation practices in particular do not mean anything to them. If instead of focusing on software, we focus on what size could mean to these projects, we interpret size as the number of preferences to be set. Effort can mean the effort involved in setting all the preferences for a COTS product. Again, projects must write procedures to describe the process they use to derive these estimates and thus satisfy the CMM Software Project Planning goal for estimating.

An organization that provides services to a client organization runs into anomalies at every KPA in the CMM, and estimation in the Software Project Planning KPA is no exception. There are actually two environments for which processes need to be defined by the service organization trying to achieve Level 2: the service environment at the service organization and the development environment at the client site for which resources are being supplied. A service organization first needs to document one or more methods for generating estimates that its staff can use either in-house or at the client site. If the client does not have a documented estimation process, the staff uses the service organization's process when it estimates work products at the client site. If the client site has estimating procedures in place, the client might require the service organization staff assigned to the client site to follow the client's estimation process. The service organization has to provide for this contingency in its own process and policies by documenting the option of deferring to the client's process when the client has a process in place that is at least as comprehensive as its own. If the client's process is documented but less comprehensive, it might be necessary to augment that process with the missing pieces from the service organization's process to maintain estimation consistency across all projects in the service organization.

Schedule Development

We are all familiar with schedule-driven projects, where an aggressive marketing group, eager to please the customer, makes delivery promises without input from the software organization. In other cases, management sets project schedules with little or no input from the technical staff, or market demands control delivery dates, when necessary to beat the competition to market, participate in a trade show, or meet mandatory calendar-year product upgrades. A show of hands at a recent meeting indicated that only three out of 70 attendees were working with schedules on which they had some input. Unfortunately, when the technical software staff is not part of the scheduling process, the delivery date is not always based on reality. A project might be physically unable to meet its deadlines no matter how many resources it applies to the project, or the organization might not have enough available resources to accomplish the project's work in the allotted time without impacting other projects.

Not surprisingly, CMM practices state that individuals from the software group should participate in the project proposal effort. Some organizations have put alternative practices in place to make this happen. One organization gives marketing a checklist of the conditions under which marketing can make commitments to a customer without the software group's direct participation. The checklist lays out well-defined, narrow limits on the type of product, the organization's experience level with the product, and the marketing person's knowledge of the product as conditions to be met before marketing has license to make delivery or product commitments. If any one of the checklist conditions is not met, marketing must involve the software group to approve any commitments that affect that group. A different organization lets marketing give customers initial rough estimates for cost and schedule, which it finalizes under contract only after the software group defines the requirements in detail and software and other affected groups have agreed to the final estimates. In both of these cases, the software group is involved at some point,

Not surprisingly, CMM practices state that individuals from the software group should participate in the project proposal effort.

Related Resources

1. J.G. Brodman and D.L. Johnson, *The LOGOS Tailored CMM for Small Businesses, Small Organizations, and Small Projects*, LOGOS Int'l, Colorado Springs, Colo., 1996.
2. D.L. Johnson and J.G. Brodman, "Tailoring the CMM for Small Businesses, Small Organizations, and Small Projects," *Elements of Software Process Assessment and Improvement*, K. El Emam and N.H. Madhavji, eds., IEEE Computer Soc. Press, Los Alamitos, Calif., 1999.
3. M.C. Paulk, "Using the Software CMM with Good Judgment," *ASQ Software Quality Professional*, Vol. 1, No. 3, June 1999, pp. 19–29.
4. M.C. Paulk, "Using the Software CMM in Small Organizations," *Joint 1998 Proc. Pacific Northwest Software Quality Conf. and Eighth Int'l Conf. Software Quality*, 1998, pp. 350–361.
5. M.C. Paulk et al., *Capability Maturity Model for Software, Version 1.1*, Tech. Report CMU/SEI-93-TR-24, Software Eng. Inst., Carnegie Mellon Inst., Pittsburgh, 1993.

whether it be in the form of originating written guidelines or in approving the final commitments. All organizations must document these alternative practices either in a policy, a procedure, or the organization's standard process for making schedule commitments on a project.

Resources on a schedule-driven project are determined by the amount of time available to accomplish the project. If the necessary resources are not available and this is the business climate under which the organization often operates, then the organization needs to define and document a process for accommodating aggressive scheduling practices. Some projects prioritize workload tasks and implement them in priority order, scoping the product capability according to the time available (similar to the LOE budgeting example that we discussed earlier). Ideally, projects define and implement a basic capability for the product first and add features as time permits so that there is always some capability to deliver to the customer. All too often, however, we see projects try to implement everything promised, only to have nothing to show the customer when the drop-dead delivery date arrives.

To put these practices in the context of the CMM, procedures and policy must reflect the organization's practices. If a project cannot start with a basic product and add functionality as time permits but instead descopes the workload as a risk-mitigation practice, then the organization must document this practice as an alternative practice for a project. Otherwise, CMM assessors might perceive a project as inadequately performing risk assessment by dropping functionality. An assessment team we recently queried did not perceive descoping functionality as a viable risk-mitigation practice. However, if the descoping is due to the organization's business climate and not poor project planning, and the organization documents the practice as an alternative practice, then the project meets the CMM's Software Project Planning estimation goal.

There are projects where schedule end dates are controlled by external demands, such as in an organization that develops software for the IRS or a state tax department, both of which have mandatory tax-year deadlines. Projects supporting these customers have fixed delivery dates, but unfortunately, the customers are often late in supplying the data the organization needs to build the software. The project's planning process, thus, must heavily rely on risk mitigation to compensate for the potentially reduced schedule, and project estimates must compensate for the potential risks.

This is a representative sampling of software project planning practices in nontraditional software development environments. There are many more examples, and there are other aspects of project planning that space and time do not let us cover. There are also the remaining 17 KPAs of the CMM that we could examine. We have tried to show that for most organizations, no matter how unique they appear to be, there is a way to package unique practices so that these practices satisfy the CMM's goals. Most CMM-challenged organizations assume that they are at Level 1 and throw their hands in the air when addressing the CMM. What they need to remember is that many of their practices are good and are suited to the organization's business environment and goals. They should base their improvements on those practices.

When considering a software process improvement program, the CMM is a good model, because it provides detailed guidance based on sound software engineering practices. However, the CMM cannot possibly fit the needs of all organizations equally and cannot be interpreted literally. An organization needs to use common sense when applying the CMM structure and enhance its own practices such that they meet the CMM's goals. If an organization does not have the foundation of the needed practices in place already, it can introduce practices that have worked for organizations and projects like theirs, rather than force-fit the CMM practices onto their organization. An organization will achieve little benefit from a software process improvement program if it is not tailored to the organization's needs and business environment.

We hope that by introducing organizations to a different approach to meeting CMM goals they will overcome their perceived inability to use the CMM as an improvement model.

References

1. C.M. Paulk et al., *Key Practices of the Capability Maturity Model*, Version 1.1, Tech. Report CMU/SEI-93-TR-25, Software Eng. Inst., Carnegie Mellon Institute, Pittsburgh, 1993.
2. *Improving the Software Development Process*, training course, LOGOS Int'l, Colorado Springs, Colo., 2 Sept. 1999.

About the Authors

Donna L. Johnson is president of LOGOS International, a Boston-area software management consulting company. In addition to consulting, she conducts training and performs appraisals. She is former chair and current SEI contact for the Boston SPIN organization, of which she is a cofounder. She has a BA in mathematics from Wheaton College, Norton, Massachusetts, and an MA and MS in numerical science from Johns Hopkins University. Contact her at LOGOS Int'l, 8 Mackintosh Ln., Lincoln, MA 01773; johnson@logos-intl.com.

Judith G. Brodman is CEO and cofounder of LOGOS International. Her research interests include the ROI calculation for software process improvement, the adaptation of the CMM to diverse organizations, and the identification of management and quality metrics and software risk and management methods. She has taught software project management at Northeastern University, and she received her MS in computer engineering from Boston University and her BS in mathematics from Emmanuel College. She is a member of the AFCEA, the IEEE, and the Massachusetts Software and Internet Council. She is an editorial board member for *Software Process Improvement and Practice*.