MIS QUARTERLY EXECUTIVE

ENABLING PROCESS DISCIPLINE: LESSONS FROM THE JOURNEY TO CMM LEVEL 5[1]

Paul S. Adler
University of Southern California

Frank E. McGarry
Computer Sciences Corporation

Wendy B. Irion-Talbot
Computer Sciences Corporation

Derek J. Binney
Computer Sciences Corporation

Executive Summary

A growing number of organizations are using the Software Engineering Institute's (SEI) Capability Maturity Model® (CMM®) to improve their software development process. Following a Total Quality Management philosophy, the CMM defines five successively higher levels of "process maturity," each requiring progressively more discipline in development. Some critics believe such organizational discipline burdens and demotivates developers. Computer Science Corporation's experience, however, has been more positive.

This article presents lessons that two Computer Sciences Corporation units learned in their journey to Level 5. Interviews with developers and managers found that although the higher levels of CMM imposed considerable paperwork burden, team members usually experienced the resulting discipline as enabling rather than coercive. We found four key factors needed for successful CMM implementation: creating sufficient strategic impetus to pursue CMM certification, sustaining management commitment to the requisite investments of time and resources, orchestrating broad staff participation in defining and refining processes, and facilitating organizational socialization to encourage software developers' buy-in to CMM-style discipline.

WHY AIM FOR CMM LEVEL 5?

Software development, in particular, the development of large-scale systems, is still often chaotic, with too many programs failing entirely or delivered late, over budget, and with poor quality.[2] In 1984, the U.S. Department of Defense (DoD) funded the Software Engineering Institute (SEI) to create a model of a more reliable software development process.[3] With considerable industry assistance, SEI developed the Capability Maturity Model (CMM®). It was inspired by Total Quality Management principles and defines five progressively more "mature" forms of the software development process, from Level 1 (ad hoc), through repeatable, defined, and managed, to Level 5 (optimizing).[4]

The CMM is now one of the most popular means for improving software development. The first appraisals of organizations using the CMM took place in 1987; by the end of 2003, over 2000 organizations had been appraised, some more than once. During 2003 alone, SEI conducted over 450 appraisals, including a growing number outside the U.S..

The CMM has moved far beyond the U.S. DoD world. Of the 1,342 organizations appraised between 1999 and June 2003, 74% were in the commercial sector, 22% in U.S. DoD or federal contractors, and 4% in military or federal government. Its popularity is not confined to software services consulting firms: most of its users are in-house IS groups. Its applicability does not depend on the traditional waterfall approach to development: it is compatible with iterative development. Nor does it require an army of support staff: it has been implemented in both large and small organizations. Over one-half the appraisals have been in

[1] Carol Brown was the accepting Senior Editor for this article.

[2] Standish Group Chaos study report, 2003, available at www.standishgroup.com; Gibbs, G.G., "Software's Chronic Crisis," Scientific American, September 1994, p. 86; Lieberman, H. and Fry, C. "Will Software Ever Work?" *Communications of the ACM,* 2001, 44, 3: 122-124; Jones, C., 2002, "Defense software development in evolution," *CrossTalk*, November, 26-29.

[3] Humphrey, W.S., "Three process perspectives: Organizations, teams, and people," *Annals of Software Engineering*, 2002, 14, pp. 39-72.

[4] Appendix 1 provides more information on the CMM.

organizations under 100 people, and over 28% in organizations under 50 people. In fact, many of the CMM's recommendations can be implemented by individual teams, or even individual developers.[5]

Evidence is accumulating that achieving higher levels of CMM process maturity does indeed improve development performance in quality, cost, and timeliness. Lockheed Martin's space shuttle unit in Houston, for example, took five years to go from Level 1 to 5, and over that time it decreased defects by 90%, cycle time by 40%,[6] and development costs by 40%. Several multi-organizational studies also support the thesis that higher CMM levels improve productivity. According to one such study, total development costs decreased by 5% to 10% (depending on size of project) as each higher level was attained.[7]

However, the CMM also has its critics. The most common criticism is that a more mature process requires too much documentation and burdens and constrains software developers. Under the CMM, developers lose much of their traditional autonomy. Their control and discretion in deciding which methods to use are greatly reduced because these methods are largely standardized and formalized. Critics thus worry that developers' motivation will suffer, and that given the importance of motivation to long-term effectiveness in software development, the CMM will ultimately have negative consequences on both the human and the economic planes.[8] Two well-respected software management experts summarized their concerns in these terms:[9]

> "Of course, if your people aren't smart enough to think their way through their work, the work will fail. No Methodology will help. Worse still, Methodologies can do grievous damage to efforts in which people are fully competent. They do this by trying to force the work into a fixed mold that guarantees a morass of paperwork, a paucity of methods, an absence of responsibility, and a general loss of motivation."

One project manager interviewed for the present study summarized the issue this way:

> "Programming has always been seen as more of an art form than a factory process. Programmers are supposed to be creative, free spirits, able to figure things out themselves. So the software factory idea was very alien to the culture of programmers."

Critics of the CMM are much more comfortable with "agile programming" approaches, such as extreme programming and feature-driven development, which promise process improvement without bureaucracy. However, as Boehm and Turner argue, agile methods have proven appropriate only where systems and development teams are small, where customers and users can consult each other frequently, and where requirements and the environment are very volatile.[10]

What, then, about the rest of the software world, where a more plan-driven approach is feasible and indeed essential? Is this huge part of the software development industry doomed to suffocate under stifling bureaucracy? Does the CMM merely sanctify this sad destiny?

Computer Science Corporation's (CSC) experience implementing the CMM has been a happier one. The main lesson learned is that the effect of CMM-style process discipline depends on how it is implemented. Discipline can be enabling; it does not have to be coercive and burdensome. The CMM can be implemented in ways that support developers' motivation and performance. Thus, the CMM can provide a path for process improvement, especially for organizations with larger projects, using larger teams, where customers or users are not collocated with developers, and where requirements may be changeable but are not so volatile as to defeat all planning. At this time, when more and more large organizations are shifting to multiple providers – both in-house and outsourcers – CSC's experiences provide lessons for a broad range of software development organizations.[11]

---

[5] See, Humphrey, W. S., PSP (sm): *A Self-Improvement Process for Software Engineers*, New York: Addison Wesley Professional, 2005; and. Humphrey, W.S., *Introduction to the Team Software Process (sm)*, New York: Addison Wesley Professional, 1999.

[6] Gibbs, G.G., "Software's Chronic Crisis," *Scientific American*, September 1994, p. 86.

[7] Clark, B., "Effects of Process Maturity on Development Effort," unpublished paper, May 1999, based on his dissertation, available at http://sunset.usc.edu/~bkclark/Research. Multi-organization studies include: Krishnan, M.S., Kriebel, C.H., Kekre, S., and Mukhopadhyay, T., "Productivity and Quality in Software Products," *Management Science*, 46, 6, June 2000, pp. 745-759; Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., and Paulk, M., "Software Quality and the Capability Maturity Model," *Communication of the ACM*, 40, 6, June 1997, pp. 30-40; and Harter, D.E., Krishnan, M.S., and Slaughter, S. A., "Effects of Process Maturity on Quality, Cost and Cycle Time in Software Product Development" *Management Science*, 46:4, 2000, pp. 451-466.

[8] Conradi, R. and Fuggetta, A,, "Improving Software Process Improvement," *IEEE Software*, July/August 2002 92-99. Crocca, W. T., "Japan's Software Factories: A Challenge to U.S. Management," *Administrative Science Quarterly*, 1992, 37, 4: 670-674; Lynn, L. H., "Assembly Line Software development," *Sloan Management Review*, 1991, 88: Bach, J. "The immaturity of CMM," *American Programmer*, 7, 9, 1994, on-line at http://www.satisfice.com/articles/cmm.htm.

[9] DeMarco, T., and Lister, T., Peopleware: *Productive Programs and Teams*, New York: Dorset, 1987, p 116..

[10] Boehm, B, and Turner, R., *Balancing Agility and Discipline*, Boston: Addison-Wesley, 2003, p. 22.

[11] Other case studies of Level 4 and 5 organizations are presented in: Special issue on CMM Level 5 at the Ogden Air Logistics Center, *CrossTalk*, May 1999, 12, 5; Dutta, S. and Van Wassenhove, L., "Motorola India and Excellence: Life beyond CMM Level 5," INSEAD

This article is based on our analysis of archival material and 32 interviews with employees at all organizational levels in two Level 5 CSC "programs." (Programs at CSC are organizational units devoted to long-standing, multi-project client engagements; they are therefore more similar to in-house MIS groups than to consulting organizations with continually changing clients.) Both programs are in CSC's Federal Sector organization, which serves the U.S. government exclusively.[12]

CSC'S PROCESS IMPROVEMENT EFFORTS

CSC is one of the largest professional software services firms in the world. In 2002, the time of the study, it had annual sales of $11.4 billion and employed 67,000 people.

In the company's process improvement efforts over the past decade, each program was free to adopt the CMM or not. In practice, many government sector programs (including Programs A and C) found that more and more of their customers expected their contractors to be certified at CMM Level 3. Program A adopted the CMM relatively early, driven primarily by the program manager's conviction that it would give his unit long-term competitiveness by guiding its aggressive process improvement efforts. Program A was first assessed at Level 1 in 1991; it took until 1996 to reach Level 3, and until 1998 to reach Level 5. Program C had long operated under the U.S. DoD's military standards for software quality; the CMM progressively supplanted these standards as customers shifted to it. Program C was evaluated at Level 4 in mid-1998, and Level 5 in 2001.

case study, 1998; Humphrey, Watts S., Snyder, T.R., and Willis, R.R., "Software process improvement at Hughes Aircraft," *IEEE Software*, 8, 4, July 1991, 11-23; Diaz, M. and Sligo, J., "How Software Process Improvement Helped Motorola," *IEEE Software*, 14, 5, Sept-Oct 1997, pp. 75-81; Pitterman, B., "Telecordia Technologies: The Journey to High Maturity," *IEEE Software*, July-Aug. 2000: 89-96; Keeni, G., "The Evolution of Quality Processes at Tata Consultancy Services," *IEEE Software*, July-Aug 2000, pp. 79-88; Wigle. G. B. and Yamamura, G., "SEI CMM Level 5: Boeing Space Transportation Systems Software," in G. Gordon Schulmeyer, James I. McManus, eds, *The Handbook of Software Quality Assurance*, 3rd ed., page 351-380, Upper Saddle River NJ: Prentice Hall PTR, 1999; McGarry, Frank and Decker, William, "Attaining Level 5 in CMM Process Maturity", *IEEE Software*, November/December 2002, pp. 87-96; Butler, K. and Lipke, W., "Software Process Achievement at Tinker Air Force Base, Oklahoma," CMU/SEI-2000-TR-014; and Willis, R.R., Rova, R.M, Scott, M.D., Johnson, M.I., Moon, J.A., Shumate, K.C., and Winfield, T.O., "Hughes Aircraft's Widespread Deployment of a Continuously Improving Software Process," SEI Technical Report CMU-SEI-98-TR-006. Interested readers will also find the reports on SEI's "High-maturity workshops" useful; they were conducted in 1999, 2000, and 2001, and are available at www.sei.cmu.edu,

[12] Details on our research methods are provided in Appendix 2, and information about the two Level 5 programs (Programs A and C) are presented in Appendix 3.

The Process Discipline of the CMM

How constraining are the formalized, standardized processes recommended by the CMM? And how much does their discipline impinge on developers' autonomy? The short answer: a great deal.

Like other large organizations, CSC deployed a hierarchy of controls:

1. Policies defined universal, corporate requirements;
2. Processes defined the methodologies used by programs;
3. Procedures defined activities within a program; and
4. Work Instructions defined requirements at the individual task level.

The work instructions used in Program C illustrated the fine-grained specificity of the resulting process discipline. The program had separate work instructions covering such tasks as high-level design, each of two types of low-level design and two types of code reviews, testing, change request implementation, change request resolution, and root cause analysis. Each instruction was several pages long, often specified forms to be filled in and showed flowcharts to describe the sequence of steps involved. In the past, the binders describing these processes took up some eight feet of shelf space. Now, this documentation is on-line, and work-flow procedures are increasingly built into automated collaboration systems.

To some extent, this high level of formalization was due to the nature of these programs' tasks. The typical software products in Programs A and C were relatively large (see Appendix 3). Moreover, government clients often required significantly more documentation than commercial sector clients. Furthermore, the life-threatening risks associated with Program A's and C's products encouraged formalization. In these respects, both organizations were somewhat atypical software development organizations; but for precisely this reason, they illustrate how staff respond to high levels of process discipline.

How the Staff Responded

Prior to our own investigation, several metrics suggested that Program A and C staffs mainly supported the CMM initiatives. First, both programs had little difficulty attracting or retaining staff, even with the rigorous discipline of Level 5, with the talent shortage at the time, and with the relatively modest compensation levels they offered. Second, a 1999 internal survey of Program A staff conducted after the program had reached Level 5 showed that the staff were largely positive. Of those respondents who had participated in

a CMM audit by external evaluators, 79% thought the CMM was "well worth the effort" and another 18% thought it was "of some value." Of those who had not participated in an audit, 58% saw the CMM as "well worth the effort" and another 30% as "of some value." Only 3% of the audited staff and 12% of non-audited thought it was "of little or no value."

In our own interviews, we uncovered two main concerns about the CMM. But these concerns were counter-balanced by five positive features the staff appreciated in the new way of working. The first concern was the amount of documentation. Several staff members made comments such as these:

> "I understand why we need the CMM evaluations. But it's added a lot to the amount of documentation we need and the number of interviews we have to go through. I suppose that in the long term, this documentation might help us improve, but for the developers, it's added a lot of paperwork." (C: developer)

A second concern was the need for some of the Levels 4 and 5 requirements. Many interviewees saw their merit, but some were more skeptical:

> "We struggled to get past Level 3. Level 3 seems to give you most of CMM's benefits. Frankly, Levels 4 and 5 haven't changed or helped much. Beyond 3, documenting the technology management process didn't really do much for us: we manage to change technology pretty effectively without formalizing that process. But on the other hand, defect prevention has been very useful." (A: contract officer)

> "I think Level 3 was worth doing. But most of Levels 4 and 5 just don't seem to add much. It isn't about everyday stuff anymore. We are doing most of these processes, and documenting them adds a lot of cost but not much value." (C: project manager)

Counterbalancing these concerns, the development staff mentioned six benefits of the CMM and process maturity. The CMM:

1. Provides a scaffolding for organizational learning
2. Improves quality
3. Facilitates coordination
4. Rationalizes paperwork
5. Does not inhibit creativity
6. Supports fact-based management.

Provides scaffolding. Education theorists use the metaphor of *scaffolding* to describe the temporary assistance a teacher provides a student who is trying to master a new task.[13] The CMM functioned as scaffolding for organizational learning in these two CSC units. Even those who began with doubts about some Level 4 or 5 KPAs often came around to seeing their value:

> "I found the quantitative process management KPA a difficult concept to understand, at least at first. But after a while, I came around to seeing the value of that KPA. For example, a while ago, we ran into a schedule problem during testing, and we did the usual thing: add testing resources to regain schedule. But it still didn't work. So I asked the testing organization to look at their process, and we found that it was very labor intensive, requiring a lot of documentation that was rarely needed or used. We worked out that if the documentation was needed, it was more cost-effective to rerun the test. So testing streamlined their process. It was too late to regain the lost schedule, but we did prevent further erosion and it will help us on the next program." (A: project manager)

> "Writing the process down has had some great benefits. It's made us think about how we work, and that's led to improvements. For example, formalizing the training program has helped bring some outliers into conformance. And we formalized the SEPG (Software Engineering Process Group) process, and that has helped stimulate improvement." (C: training)

Improves quality. While the formalized process did reduce developers' autonomy, the sense of personal loss was often outweighed by pride in improving the quality of the resulting product:

> "I didn't believe in all this process stuff at first. I was like most developers — all the paperwork seemed ridiculously burdensome. But now I can see the critical value of it. I wouldn't have said this ten years ago, but now I see the value of metrics. They have to be the right ones, and not too many; but they really do help us identify problems and get more effective. For example, take our problem reports. Anyone can write one. The technical leads review them and we track them by category. That allows us to detect

[13] See Wood, D., Bruner, J., and Ross, G., 1976, "The Role of Tutoring in Problem-Solving," *Journal of Child Psychiatry and Psychology*, 1976, 17, 89-100.

underlying, hidden problems — instead of focusing on individual deficiencies. [...] Developers want above all to deliver a great product, and the process helps us do that. What I've learned coming here is the value of a well thought-out process, rigorously implemented, and continuously improved. It will really improve the quality of the product. In this business, you've got to be exact, and the process ensures that we are. You have to get out of hacker mode!" (A: developer)

Facilitates coordination. Process discipline helped departments coordinate tasks within and among themselves:

"In a small organization doing small programs, you have a lot of flexibility, but there's not much sharing. You're kind of on your own. Here, I'm just a small part of a bigger program team. So you don't do anything on your own. It's a collaborative effort. So there has to be a lot of communication between us. And the process is there to ensure that this communication takes place and to structure it. The process helps keep us all in sync." (C: developer)

Rationalizes paperwork. While documentation did increase with process maturity, maturity also allowed an organization to be document more selectively, thereby rationalizing the paperwork. Testing illustrated this benefit:

"Over the last ten years, we've refined the test procedures considerably. First, we have better tools. Documentation and reports that used to take two or three days each week to create can now be generated in an hour. Second, we streamlined some of the procedures for some programs. Now we have a generic template, which we can modify to suit the circumstances. We moved from prescribed, detailed test tables to simpler and voluntary guidelines based on historical examples. And with the benefit of experience and analysis, we are collecting more useful information and less of the kinds of information that proved to be not all that useful." (A: tester)

Does not inhibit creativity. Overall, process discipline helped, rather than hindered, creativity:

"Even when tasks are more innovative, you can still get a lot of advantage from process. You need to sit down and list the features you want in the system, and then work out which are similar and which are different from your previous work. And you need to make sure the differences are real ones. You'll discover that even in very innovative programs, most of the tasks are ones you've done many times before. Then, for the tasks that are truly novel, you can still leverage your prior experience by identifying somewhat related tasks and defining appropriate guidelines based on those similarities. They won't be precise work instructions of the kind you'll have for the truly repetitive work, but these guidelines can be a very useful way to bring your experience to bear on the creative parts of the work." (A: tester)

Supports fact-based management. Process maturity discouraged autocratic management styles, as well as the gaming associated with autocratic styles:

"Now, when you sit down with your manager, there's a lot more data on the table. It used to be much more a matter of gut feel — 'I'm worried we're not going to make it.' And then it was their opinion against yours. Now your manager is going to say, 'I've been tracking our progress against our detailed schedule estimate. I'm concerned that there's a bunch of units not done yet. What can we do to resolve this?'" (A: systems engineering)

CMM SUCCESS FACTORS AND PITFALLS

In analyzing these positive responses to process discipline, we identified four key success factors, which form a causal chain. Each link is the prerequisite for the next, and presents its own challenges.

1. Creating strategic impetus to pursue CMM certification
2. Sustaining management commitment to investments in process improvement
3. Orchestrating broad participation in defining and refining processes
4. Ensuring organizational socialization to encourage developer buy-in for process discipline.

The following paragraphs discuss these four success factors in turn, identifying how each contributed to the broad pattern of success in these two CMM implementations, along with the associated risks and pitfalls, and some countermeasures.

Creating Strategic Impetus for CMM Certification

The overall impetus behind the CMM certification efforts in these CSC programs was senior management's conviction that both the capabilities it would help develop and the official recognition of these capabilities would provide competitive advantage. Externally, high-level CMM certification would reassure potential customers and qualify the program to bid on government contracts. Internally, the proponents of process discipline argued that CMM-style process improvement could help CSC build capabilities crucial to its success. Surely, if the managers believed that the CMM recommendations were technically unsound, then the external pressures for certification would have had very different effects. Likewise, without the external pressures, the proponents would have had a more difficult time mobilizing and sustaining the strong executive-level commitment needed to pursue process improvement.

CSC customers who supported its process efforts were motivated by, among other things, the visibility and control they would gain from the high process maturity:

> "One of the main forces behind process here has been the customer, who wanted more visibility into the progress of each program. And you can understand their motivation. If you were building a house, you'd want your contractor to be able to give you precise status reports. It's the same in software." (C: project manager)

CSC's customers believed CMM certification signaled that the organization was able to assure a faster start-up of work because the requisite practices and infrastructure would already be in place. They would have access to documented performance records, they would be reassured that management viewed quality as a high priority. and, as a result, they would feel more confident that CSC's program targets would actually be met.[14]

On the other hand, sometimes customers' recognition of the importance of process maturity oscillated with changes in their own organization.[15] Faced with this volatility, CSC found it difficult, if not impossible, to build process maturity:

> "If I compare Program C with others that I've seen, the importance of customer maturity is obvious. With a low-maturity customer, we'll see a high volume of change that's not managed well. Changes in direction and priority are the norm. The daily flux will be so great, and the demands so extreme, that it's very difficult for us to package changes so that releases of the product can be controlled and tested. Our internal controls work effectively when we're given time, but when time or priorities are changed by the client (who is always right — by definition), it creates a very challenging environment." (C: process engineering)

One way CSC overcame this hurdle was to invite customer representatives to weekly review meetings. Another approach was to translate process issues into "bottom line" implications for customers:

> "Our customer has been rated a CMM Level 4, but they don't always seem to implement their process. For example, in one of our programs a while ago, the requirements kept changing and the scope kept growing, but the customer wasn't following a disciplined process for controlling these changes and just didn't want to hear about the implications. The requirements kept drifting so much that it was very hard to even regularly update our estimate of the size of the program. They just ignored our concerns for nearly a year. Finally we issued a cost report that showed that that we'd need 25% more staff-months. Putting it in dollars finally got their attention. But not before we had wasted a lot of work and time." (C: project manager)

The difficulty, of course, is that only development organizations that are already at high maturity levels are able to quantify the costs of immaturity. So there appears to be a limit to how much more mature than its customers a software services organization can be.

Sustaining Management Commitment to Investments in Process Improvement

Strategic impetus encouraged, but did not guarantee, the requisite management commitment. Implementing more mature processes required senior management to invest substantial time and attention to overcome organizational inertia and ensure that work practices really changed.

---

[14] Also see: Gansler, J.S., "Memorandum for Component Acquisition Executives," *Crosstalk*, January 2000; Etter, Delores, "Acquisition Software Oversight," *CrossTalk*, August 1999; and Aldridge and Stenbit, 2003, *Memorandum for Secretaries*, http://www.acq.osd.mil/SIS/Publications/804memo1.pdf.

[15] Also see Basili, V.R., McGarry, F., Pajerski, R., and Zelkowitz, M.V., "Lessons Learned from 25 Years of Process Improvement: The Rise and Fall of the NASA Software Engineering Laboratory," *Proceedings of the International Conference of Software Engineering (ICSE)*, May 2002, available at: http://www.cs.umd.edu/projects/SoftEng/ESEG/papers/83.88.pdf.

Several features of CSC's management commitment stood out. At higher maturity levels, process status — as distinct from project status — was routinely addressed in management review meetings. At all levels, management participated in process-related activities, which included guiding subordinates on the appropriate use of process, setting process goals, and reviewing a program's progress towards these goals. Management commitment was visible not only in personal participation but also in incentive and promotion policies, the way resources were allocated to process-related activities, and in management's reluctance to approve "waivers" from improvement and assessment activities.

A hallmark of CSC's high-maturity programs was the commitment of organizational resources. Staffs in specialized process engineering, quality assurance, and configuration management were regularly funded. Program A devoted some 1.75% of its total headcount to quality assurance and another 1.25% to process improvement. Program C committed similar levels, with 1.25% of total headcount for process improvement and 3.25% for quality management. More subtly, management commitment ensured that the staff dedicated to these activities were themselves exceptionally competent and well respected by their peers.

These commitments, however, had to be sustained over many years, and wavering management commitment is a dangerous pitfall for enterprises seeking to implement the CMM.[16] Under schedule and profit pressures, management often found it difficult to fund specialized staff, or to continue to live by the new discipline:

> "One key challenge is maintaining buy-in at the top. Our top corporate management is under constant pressure from the stock market. The market is constantly looking at margins, but government business has slim margins. That doesn't leave much room for expenditures associated with process improvement — especially when these take two or three years to show any payoff." (C: process engineering)

> "We could do better at capturing and using lessons learned. We have all the vehicles for doing it — presentations, newsletters, databases. But it takes time. And there are so many competing priorities. In the end, it's all about profit and meeting schedules!" (laughs) (A: project manager)

There was a constant danger that the combined pressures of the bottom line and CMM certification might encourage a disconnect between the true daily work practices and how they were described to the outside world:

> "I can see that external evaluations are a very important learning tool. It's just like in college: 90% of what the student learns is in the week before the test! So we do need the test to create that incentive. But it's not an end in itself. The real issue is: Is passing the test just a veneer? That depends on how the managers treat the test – as an opportunity to put some banners on the walls, or as an opportunity to focus attention and get some real learning done. At Program A, we have reached (well, almost reached) the point where people like the tests as an opportunity to show off their improvements." (A: systems engineering)

Under cost pressures, it was difficult to fund the investment needed:

> "We do ask program teams to do a Lessons Learned report at the end of the program. We post the results on the database. But there's no staff support for the process." (A: quality assurance)

> "There's no doubt that more process maturity means more paperwork. Some of it is good, and some of it is an impediment, especially from a productivity point of view. Unless we have the tools to automate this documentation, it has to slow us down. We still don't have the right tools." (C: program manager)

Executives have faced such challenges for over a century. When Frederick Taylor developed his revolutionary "scientific management" techniques for bringing discipline to the manufacturing shop-floor, the greatest impediment he encountered was not worker or union opposition but management's reluctance to make the requisite overhead investments.[17]

To keep higher-level executives committed, CSC's process champions continually reinforced the notion that process improvement and CMM certification were investments, not expenditures. Once their organizations had reached CMM Level 3 – the level required by many customers – champions argued that further investment in process would yield big returns by streamlining and automating the process to alleviate the documentation burden, by tailoring it to further

[16] Also see Verner, J.M., Overmyer, S.P., and McCain, K.W., "In the 25 Years Since *The Mythical Man-Month*, What We Have Learned About Project Management," *Information and Software Technology*, 1999, 41(14), 1021-1-26.

[17] Nelson, D., *Frederick W. Taylor and the Rise of Scientific Management*, Madison: University of Wisconsin Press, 1980.

reduce the burden for small-scale, low-risk tasks, and by accelerating learning across projects.[18]

Orchestrating Broad Participation in Defining and Refining Processes

Given sufficient strategic impetus and management commitment, the third key success factor was staff participation. Both Level 5 programs had formalized participation processes and worked hard to propagate a leadership style that was participative rather than autocratic

The development process under CMM was highly standardized and formalized, but it was not imposed from above by an isolated staff function. Rather, it was defined and refined in a highly participative manner. Each program was free to tailor the corporate standard process (see Appendix 3); and within programs, each project was also encouraged to tailor the standard process to its needs. This latter tailoring effort was organized under the Process Assurance Cycle (PAC):

> "We have 101 S&Ps [Standards and Procedures] that embody our documented processes and our development methodology. In the PAC process, the program manager can add, delete or modify these to suit the needs of the program. We don't track it formally, but I'd guess that of the average program's S&Ps, about 40% are the standard S&Ps adopted without change, about 20% are modified versions of the standard set, and the other 40% are written by the program for use by the program. The modifications and new S&Ps are often driven by new technologies like the Web or new languages like C++. But we also modify them if the program is unusually small or large, or if the customer has unusual requirements The PAC process gives the program managers the option of tailoring our S&Ps to the needs of their program. Typically, a senior person within a program will draft the tailored S&Ps and then give that draft to the people in the group to review. That helps ensure that the S&Ps reflect real-life concerns and that we get buy-in." (A: process engineering)

More generally, Programs A and C actively encouraged participation in a large range of improvement projects. In Program C, for example, every subunit participated in the process change management process orchestrated by the software engineering process group (SEPG); and staff across the organization were encouraged to submit process improvement ideas:

> "My manager is very open to suggestions for process improvement. He wants us to document our suggestion, but then he'll be pretty aggressive about taking it forward to the SEPG. And then we'll usually get a message from the SEPG telling us what's happening with the idea -- although sometimes the closure doesn't happen, and that's a bit frustrating. If the suggestion is a local one, just for our group, we usually have a little budget to explore it and implement it if it proves to be a good idea. For example, we decided it would be useful if we had a particular online request form. So we implemented a system for capturing all these requests, then we could list them out for approval for submitting to the customer. We easily got a green light to go ahead with that." (C: developer)

When we asked interviewees which factors led developers to adopt negative attitudes toward CMM-style discipline, many noted lack of opportunities to participate. Specifically, they mentioned managers not explaining the rationale behind some process requirements, not involving staff in process definition, and not responding to subordinates' suggestions for improvement.

Given the importance of obtaining broad participation, CSC put considerable weight on leadership style when selecting and developing its managers. However, admonitions about the desired style can easily be blunted by the many other pressures that weigh on managers. These programs demonstrated their commitment to participation by transferring or letting go several managers whose alignment with this value was insufficient:

> "By and large, we haven't had too much difficulty bringing our managers around to this more collaborative approach. But we choose our project managers with an eye to their commitment to collaboration, too. We did have a problem with one staff person. He had a very difficult relationship with the project people he was supposed to be helping. We got a lot of complaints that he was trying to force the projects to conform to his idea of how they should function. We tried to counsel him and get him to work in a more cooperative way. But he just wouldn't ease up. Eventually we just had to let him go. And we had quite a battle with one program manager when he wasn't picked to head a new project: we felt he just wasn't

[18] Also see Diaz, M. and King, J., "How CMM Impacts Quality, Productivity, Rework, and the Bottom Line", *CrossTalk*, March 2002.

enough of a team manager. We didn't initially have any questions on the employee survey about your boss. Frankly, people were worried that managers might retaliate. But now we do, and we find the data very useful in surfacing management problems. The earlier rounds of the survey did show some big communications problems in some groups. Counseling often helped, and in some cases, we moved people out to other positions."." (A: program manager)

Facilitating Organizational Socialization to Encourage Buy-in

Perhaps the most important and novel lesson we identified in our research was that successful CMM implementation depends on the organizational socialization of the development staff. Organizational socialization – learning and internalizing the customs, attitudes, and values of the organization – was essential because so many developers coming into CSC brought with them a "self construal" – a sense of who they were – that was very individualistic and thus impossible to reconcile with the collective discipline of high maturity.

Self-construal is the constellation of thoughts and feelings people have about their relationship to others and to themselves as distinct from others.[19] Traditionally, the self-construal of developers has been strongly independent; they have low social needs and prefer autonomy in their work.[20] At CSC, such developers are called "hackers," and CSC certainly has had its share of people that fit this image.[21] In our interviews, we were therefore surprised to hear developers describe how the experience of a more mature process changes these psychological characteristics.

Several eloquently described how their self-construal had shifted from that of an independent hacker to that of an interdependent team member. This excerpt is representative:

> "Where I used to work before I came to CSC, the development process was entirely up to me and my manager. What I did, when I did it, what it was going to look like, when it was done, and so forth, was all up to me. It was very informal. Here, everything is very different. It's much more rigid. It's much more formal. A lot of people lay out the schedule, the entire functionality, and what I'm going to be accountable for — before I even get involved....
>
> When I got here I was kind of shocked. Right off, it was 'Here are your work instructions.' 'So what does this tell me?' 'It tells you how to do your job.' I thought I was bringing the know-how I'd need to do my job. But sure enough, you open up the work instructions, and they tell you how to do your job: how to lay the code out, where on the form to write a change request number, and so on. I was shocked.
>
> But I can see the need now. Now, I'm just one of 30 or 40 other people who may need to work on this code, so we need a change request number that everyone can use to identify it. It certainly feels restrictive at first. They explained the work instructions and the whole Program C process to us in our orientation seminar, but it's hard to see the value of it until you've been around a while. Now I can see that it makes things much easier in the long run.
>
> I hate to say it. As a developer, I'm pretty allergic to all this paperwork. It's so time-consuming. But it does help. You've got to keep in mind, too, that by the time we see the work instructions, they've been through a lot of revision and refinement. So they're pretty much on target." (C: developer)

This shift in attitude from independent (an "I" orientation) to interdependent (a "we" orientation) came not only through individual changes of heart but also through new recruiting and selection procedures.[22] As a result, the proportion of personnel who shared the new self-construal increased over time:

> "You won't fit in well here if you don't like structure, you prefer working by yourself, you don't like getting suggestions from other people, or you don't like taking responsibility for your work and for making it better." (A: project manager)

Where strategic impetus and executive commitment were strong, and where management encouraged participation, the relatively few reservations about process discipline were expressed primarily by technical gurus or people who had worked in smaller teams:

[19] Markus, H.R. and Kitayama, S., "Culture and the Self: Implications for Cognition, Emotion, and Motivation," *Psychological Review*, 1991 98, 224-253.

[20] Couger, J.D. and R. O'Callaghan, "Comparing the Motivation of Spanish and Finnish Computer Personnel with Those of the United States," *European Journal of Information Systems*, 1994, 3, 4, 285-301.

[21] "Hackers" here is used in the sense defined by Raymond, E.S. 1996, *The New Hacker's Dictionary*, 3rd Edition, Cambridge MA: MIT Press: it refers to someone who values above all creativity, competence, and autonomy -- as distinct from those who illegally break into systems.

[22] Also see Schneider, B., "The People Make the Place," *Personnel Psychology*, 1987, 40, 437-454.

> "We still have to deal with the 'free spirits' who don't believe in process. These are typically people who have worked mainly in small teams. It's true that a small group working by itself doesn't need all this process. But we rarely work in truly independent small teams; almost all our work has to be integrated into larger systems, and will have to be maintained by people who didn't write the code themselves. These free spirits, though, are probably only between 2% and 4% of our staff. We find some of them in our advanced technology groups. We have some in the core of our business, too, because they are real gurus in some complex technical area, and we can't afford to lose them. And there are some among the new kids coming in. too; many of them need convincing on this score. Most of them adapt; although. some don't and they leave." (C: process engineering)

The key to organizational socialization is to devote more attention to the "soft" side of processes (culture, leadership, strategic vision) and less to the "hard" side (tools, systems, procedures). A recent internal assessment of the cultural changes in Program C identified five key changes in values that resulted from its shift to a high-maturity culture:[23]

1. From everyone's world-view being parochial and stove-piped to their view being more enterprise-wide
2. From people focusing on their own individual tasks and deliverables to them thinking of their work as part of interconnected processes that can be statistically characterized
3. From their discouraging or tolerating change to their embracing change
4. From leadership by setting goals, allocating resources, and monitoring progress to leadership becoming more personal with the leaders actively engaged and their creating and communicating vision
5. From the heroes of the organization being the firefighters to the heroes being the facilitating leaders.

CONCLUSION

Managers of software development organizations are understandably cautious about introducing the process discipline required by the higher levels of the CMM, fearing that such discipline might cause scarce, talented staff to leave. But our findings suggest that many staff members find process maturity enabling – if it is implemented appropriately.

CSC's experiences confirm the lessons of a considerable body of literature on how to successfully implement large-scale organizational change.[24] Such change always requires deeply felt need and strong leadership commitment. Where it involves a professional workforce, it must be implemented in a highly participative manner. Successful change must become institutionalized as a new way of working. Beyond these established guidelines, our study suggests that institutionalization requires a deep change in staff members' self-construals – and that such change is indeed possible.

As noted, the two CSC programs described here fit Boehm and Turner's criteria for the kind of organization that can benefit from the CMM as compared to agile methods.[25] That is, these programs worked on larger projects using larger teams, where customers and users could not be collocated with developers, and where requirements volatility was not excessive. However, we have left aside the question of whether Level 5 is an appropriate goal for all such organizations. CSC's experience was that once Level 3 had been achieved, moving to Levels 4 and 5 appeared to be the logical next step to many stakeholders. Senior management saw that Level 3 did not specify a robust process for continually improving the software process, whereas Levels 4 and 5 did. Therefore, investing in moving to these levels appeared to be strategically important to both managers and development staff. That has proven to be the case.

ABOUT THE AUTHORS

Paul S. Adler

Paul S. Adler (padler@usc.edu) is professor of management and organization at the Marshall School of Business, University of Southern California. He began his education in Australia and moved to France in 1974. Paul received his doctorate in economics and management there while working as a Research Economist for the French government. He came to the USA in 1981, and before arriving at USC in 1991, he was affiliated with the Brookings Institution, Columbia University, the Harvard Business School, and Stanford's School of Engineering. His research, teaching, and consulting focus on organization design in

[23] Irion-Talbot, Wendy, "CSC Cultural Checklist/Instrument, Critical Factors for Success," *INCOSE International Symposium*, Las Vegas, NV, July 28 – Aug. 1, 2002.

[24] Cummings, T. G., and C.G. Worley, Organization Development and Change, Mason, Ohio: South-western, 2005.

[25] Boehm, B, and Turner, R., *Balancing Agility and Discipline*, Boston: Addison-Wesley, 2003, p. 22.

professional, R&D, engineering, and manufacturing operations.

Frank E. McGarry

Frank E. McGarry (fmcgarry@csc.com) is Director of Process Engineering for Science and Information Services (SIS) at CSC in Lanham, Maryland. He is responsible for systems processes and process improvement programs within SIS. In 1998, Frank led the process improvement efforts within the SEAS Center of CSC, which resulted in the first CSC CMM Level 5 rating. He joined CSC after having spent more than 25 years at NASA/Goddard where he was head of the Software Engineering Branch. He was one of the founding principals of the Software Engineering Laboratory (SEL), an internationally recognized software engineering research facility combining the talents of NASA/Goddard, the University of Maryland, and Computer Sciences Corporation (CSC).

Wendy B. Irion Talbot

Wendy B. Irion Talbot (wirionta@csc.com) is currently director of the process engineering and management office for CSC businesses serving the US Federal government, where she is responsible for process engineering, application of process and performance improvement frameworks and methods, development of CSC's integrated enterprise process architecture, and the measurement program. She collaborates with performance excellence-focused colleagues throughout CSC as the co-lead and founder of CSC's Global Process Improvement Community of Interest (COI), and co-lead of CSC's Capability Maturity Model COI. She is active in the CMMISM community where she served on the CMMI PDT, and she represents CSC on the Technical Advisory Board of the Systems and Software Consortium.

Derek J. Binney

Derek J. Binney (dbinney@csc.com) has been in the IT industry for 29 years. In his role as chief knowledge and technology office for CSC Australia, Derek considers emerging trends and technologies, providing advice to CSC clients on their implications and the opportunities they represent. Derek has published a number of papers on knowledge management (KM) and KM strategy and participates in a number of CSC internal and public forums. He serves as chair of the External Advisory Committee to the IT Faculty, University of Technology, Sydney and as member of the editorial advisory board for the *Journal of Intellectual Capital*. He is a frequent speaker at public and academic conferences. Derek is currently completing a doctorate on KM adoption at the Graduate School of Management, Macquarie University, Sydney.

APPENDIX 1: THE CAPABILITY MATURITY MODEL FOR SOFTWARE

The CMM distinguishes five successively more "mature" levels of process capability, each characterized by mastery of a number of Key Process Areas (KPAs).

- Level 1 characterizes an organization relying exclusively on individual skill and effort, lacking any process supports.
- Level 2 introduces processes for managing individual programs.
- Level 3 addresses the management of an organization's portfolio of programs.
- Levels 4 and 5 address how the organization quantifies and improves its processes over time.

According to the Software Engineering Institute (SEI), it takes, on average, 25 months to progress from Level 1 to 2, 23 months from Level 2 to 3, 28 months from 3 to 4, and 15 months from 4 to 5.

Figure 1 describes each level and the distribution of organizations appraised early in the life of the CMM and more recently. The underlying philosophy of this hierarchy was inspired by Crosby's five stages of Total Quality Management maturity (uncertainty, awakening, enlightenment, wisdom, and certainty).[26] The first official release of the CMM for software development was in 1991. It was subsequently complemented by CMM tools for systems engineering, people management, software acquisition, and engineering. In 2000, several of these tools were integrated into a broader tool, called CMM-Integration.[27]

APPENDIX 2: RESEARCH METHODS

With the support of senior CSC management, the first author conducted interviews with staff in four of its programs involved in government work.

[26] Crosby, P. B., *Quality is Free*, New York: McGraw-Hill, 1979. Also see Humphrey, W.S., "Three process perspectives: Organizations, teams, and people," *Annals of Software Engineering*, 2002, 14, pp. 39-72.

[27] The SEI does not "certify" any appraisal findings or maturity levels for a specific organization.

Figure 1: SEI's Capability Maturity Model[28]

Level	Focus and description	Key Process Areas	Distribution of appraised organizations in:	
			1987-1991 (132 organizations)	1999-2003 (1.343 organizations)
Level 1: Initial	**Competent people and heroics** The software process is ad hoc, occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.		80.0%	13.3%
Level 2: Repeatable	**Program management processes** Basic program management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on programs with similar applications.	* software configuration management * software quality assurance * software subcontract management * software project tracking and oversight * software project planning * requirements management	12.3%	43.5%
Level 3: Defined	**Engineering processes and organizational support** The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All programs use an approved, tailored version of the organization's standard software process for developing and maintaining software.	* peer reviews * intergroup coordination * software product engineering * integrated software management * training program * organization process definition * organization process focus	6.9%	25.6%
Level 4: Managed	**Product and process quality** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.	* software quality management * quantitative process management	0.0%	8.5%
Level 5: Optimizing	**Continuous process improvement** Improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.	* process change management * technology change management * defect prevention	0.8%	9.2%

The present study focuses on two organizations certified at Level 5: Programs A and C. The author's interviews conducted in two "sister" units at Level 3 helped put the findings in A and C in perspective, but are outside the scope of this paper so they are not presented.

Interviews in Programs A and C were conducted with a total of 32 people at varying levels of authority and in the main departments of these two programs. The interviews lasted approximately one hour. Interviewees were asked to describe their role in both the software development process and the software process improvement effort, and to compare their recent experiences with any earlier experiences in these or other organizations at lower levels of maturity. All the

[28] This table was compiled from various SEI sources: Paulk, Mark C., Bill Curtis, Mary Beth Chrissis, and Charles V. Weber, "Capability Maturity Model for Software, Version 1.1," *Software Engineering Institute*, CMU/SEI-93-TR-24, DTIC Number ADA263403, February 1993a, available at http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.024.html; and Paulk, Mark C., Charles V. Weber, Suzanne M. Garcia, Mary Beth Chrissis, and Marilyn W. Bush, "Key Practices of the Capability Maturity Model, Version 1.1," *Software Engineering Institute*, CMU/SEI-93-TR-25, DTIC Number ADA263432, February 1993b, available at: http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.025.html.

interviewees were assured anonymity; their names were replaced with identification numbers.

Interviews were tape recorded and transcribed; edited versions were sent back to interviewees for review and correction. The interview material was organized under a logical structure. The resulting draft synthesis report went to all interviewees for further comment, and their comments were incorporated in several successive versions. The present paper summarizes the key findings of this report. It was developed jointly by the four authors, with input from several other thought leaders at CSC.

Figure 2: Research methods

Role	Interviews in programs:	
	A	C
Contract officer	1	0
Department manager	1	2
Developers and testers	5	4
Process engineers	3	2
Program manager	1	1
Project managers	3	3
QA/CM staff	1	2
Systems engineering	1	0
Training	1	1
Total	17	15

APPENDIX 3: BACKGROUND ON CSC AND PROGRAMS A AND C

During the late 1980s, CSC began to collect and synthesize best practices in software development in its various programs. Its goal was to make these practices accessible organization-wide to achieve:

- reusable assets,
- transportable skills,
- risk reduction,
- consistent guidance,
- staff insulated from external flux, and
- quality and performance improvement

The result was a comprehensive, detailed, standardized development process. However, corporate leaders encouraged each organizational unit to develop its own version of the standard process, to balance its need for unit-specific adaptation with the value of cross-unit sharing. The general principle has been that lower-level units — the sector, divisions, programs, projects, and functions — are free to tailor higher-level processes, but they should be able to document and explain where they have differed from these higher-level processes.

Program A

Program A has had a continuous contractual relationship with its high-profile customer for 30 years. Historically, it has had some 20 mid-size projects under way at a time (each 100-400,000 lines of source-code). The program's tasks have become more complex as customer requirements technologies have evolved. There has recently been considerable pressure for more code reuse and tighter deadlines.

Over the past decade of process improvement efforts, Program A has seen its average effort variance reduced by over half. Average error rates have been reduced by 75% (and 50% in the last five years). Productivity has shown a consistent 6% annual rate of improvement.

Program C

Program C has also had a continuous contractual relationship with its DoD customer for 30 years, but the relationship had been mediated by other organizations. Program C undertakes two to five major projects at time, each representing about 2.5-3.2 million source lines of code. These projects created new versions of the weapons control systems they provide to the DoD.

It, too, has seen considerable improvement in productivity and quality over the past decade of process improvement efforts. The quality of its products was widely recognized: The program customer satisfaction index consistently averaged over 97%.