Center for
Effective
Organizations

LIFE AT CMM LEVEL 5: LESSONS FROM COMPUTER SCIENCES CORPORATION ON THE EFFECTIVE IMPLEMENTATION OF THE CAPABILITY MATURITY MODEL FOR SOFTWARE

CEO PUBLICATION
G 03-23 (451)

PAUL S. ADLER

Dept. of Management and Organization
Marshall School of Business
University of Southern California

FRANK E. MCGARRY
WENDY B. IRION-TALBOT
DEREK J. BINNEY

Computer Sciences Corporation

November 2003

LIFE AT CMM LEVEL 5: LESSONS FROM COMPUTER SCIENCES CORPORATION ON THE EFFECTIVE IMPLEMENTATION OF THE CAPABILITY MATURITY MODEL FOR SOFTWARE

Paul S. Adler,
Professor of Management and Organization, Marshall School of Business, University of Southern California, Los Angeles CA 90089-0808
Email: padler@usc.edu

Frank E. McGarry,
Director of Process Engineering, NASA/Programs, Computer Sciences Corporation, Lanham, Md.

Wendy B. Irion-Talbot,
Director of Process Engineering and Management, CSC Federal Sector, Computer Sciences Corporation, Falls Church, VA

Derek J. Binney,
Chief Knowledge and Technology Officer, Computer Sciences Corp. Australia, St Leonards, NSW Australia 2065

Version date: 9.12.03

LIFE AT CMM LEVEL 5: LESSONS FROM COMPUTER SCIENCES CORPORATION ON THE EFFECTIVE IMPLEMENTATION OF THE CAPABILITY MATURITY MODEL FOR SOFTWARE

ABSTRACT

A growing number of organizations are using the Software Engineering Institute's (SEI) Capability Maturity Model for Software (CMM) as a guide to improving their development process. Following a Total Quality Management (TQM) philosophy, the software CMM characterizes five successively higher levels of "process maturity" that afford progressively greater degrees of discipline in development. Some observers have expressed concern that such discipline will be experienced as demotivating by developers. This article analyzes the organizational and behavioral requirements of process maturity, and examines the proposition that the effect of CMM-style process discipline on an organization may depend on how it is implemented. We focus on four units of a large professional software services company (Computer Sciences Corporation), two of which have been evaluated at CMM Level 5 and two sister units that are at Level 3. Interviews with personnel in these units suggest that most personnel — developers and managers alike — experience the discipline of higher CMM Levels as enabling rather than coercive. We identify four key factors for successful CMM implementation: external impetus; management commitment; staff participation; socialization of developers. We also discuss some of the corresponding pitfalls and counter-measures.

LIFE AT CMM LEVEL 5: LESSONS FROM COMPUTER SCIENCES CORPORATION ON THE EFFECTIVE IMPLEMENATION OF THE CAPABILITY MATURITY MODEL

INTRODUCTION

Software development, in particular the development of large-scale systems, is still often chaotic, with a distressingly large proportion of programs failing entirely or delivered late, over budget, and with poor quality (Standish, 2003; Gibbs, 1994; Lieberman and Fry, 2001). In 1984, the Department of Defense (DoD) funded the Software Engineering Institute (SEI) to help development organizations in their search for a more reliable, more disciplined, software development process (Humphrey, 2002). With considerable industry assistance, the SEI developed the Capability Maturity Model for Software (CMM). The CMM, inspired by Total Quality Management (TQM) principles, characterizes five progressively more "mature" forms of the software development process, from Level 1, ad hoc, through repeatable, defined, managed, to Level 5, optimizing. (Appendix 1 gives more background on the CMM.)

Evidence is accumulating that higher levels of process maturity do indeed offer considerable performance advantages in quality, cost, and timeliness (Clark, 1999; Harter, Krishnan, Slaughter, 2000; Krishnan, Kriebel, Kekre, Mukhopadhyay, 2000; Herbsleb, Zubrow, Goldenson, Hayes, Paulk, 1997; McGarry and Decker, 2002). And there appears to have been movement in the industry towards higher CMM levels and greater process discipline (SEI Update, 2002). However, concern is often voiced that a more mature process will impose an excessive burden of documentation and constraint on software developers (see SEI website for bibliography of critical articles).

In this article, we focus on one particularly pervasive concern: many commentators worry that under a more mature process, developers lose much of their traditional autonomy in deciding the methods of work — since these methods are now largely standardized and formalized — and that as a result, motivation must suffer (e.g. Crocca, 1992; Bach, 1994; Conradi and Fuggetta, 2002; Lynn, 1991). Given the importance of motivation to long-term effectiveness in knowledge-intensive activities such as software development, critics fear that the CMM will ultimately have negative consequences on both the human and the economic planes.

Typical of opposition to formalized process and methodologies is this assessment by two well-respected software management experts:

> "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." (DeMarco and Lister, 1987, P. 116)

One interviewee in the present study (A-6) expressed the concern 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."

The proposition explored in the research we report here is that the effect of CMM-style process discipline on developers depends on how it is implemented. The discipline of the prison and the discipline of the orchestra will arguably elicit very different responses. This possibility has not been the subject of scientific investigation. The CMM framework itself does not specify an implementation approach — it specifies the "whats" but not the "hows" of process maturity.

As a step towards a better understanding of the organizational and human aspects of process maturity, this article presents some lessons from four "programs" (organizational units) of a large software services firm, Computer Sciences Corporation (CSC). All four programs are in CSC's Federal Sector organization, which services the US government exclusively: two programs (we will call them A and C) had recently been rated CMM Level 5, and two sister programs (B and D) at Level 3. Since our research addresses relatively under-explored issues, we did not want to rely on surveys, but instead conducted in-depth interviews with 68 development staff and managers in these units. We found that process maturity was usually experienced by developers and managers alike as enabling rather than coercive. We identified four factors for successful CMM implementation, and four corresponding pitfall and counter-measures. Before presenting these results, we set the stage with a brief description of CSC. (Appendices describe in more detail the CMM framework, our research methods, and the four programs.)

CSC'S PROCESS MATURITY EFFORTS

CSC is one of the largest professional software services firms in the world. At the time of the study, in 2002, CSC's total sales were $11.4 billion, and it employed 67,000 people. CSC experienced annual double-digit revenue growth over each of the preceding seven years. Today, CSC employs approximately 90,000 employees.

During the late 1980s, CSC began to collect and synthesize the lessons learned in its various programs with the aim of making these best practices accessible throughout the organization. The result was a comprehensive, detailed, standardized development process. The effort has continued and expanded over time. The benefits sought — and largely achieved — include:

- reusable assets — within a given organization, similarities across programs were identified, and code and documents have therefore been reused;
- transportable assets — since programs are managed similarly across organizations, each has benefited from the others' experience and used the others' code and documents;
- transportable skills — since programs are managed similarly, individuals have moved more easily between them;
- risk reduction — by relying on proven processes and code, risks to schedule, cost and quality have been reduced;
- consistent guidance — by standardizing process, including training materials, guidance given to staff has reflected best practice and is less idiosyncratic;
- insulating staff from external flux — a more disciplined process has reduced the frequency of changes in customer requirements, and thus has improved effectiveness and predictability of the development process.
- quality and performance improvement – the programs at Level 5 have observed and documented sizeable improvements in quality (decreased defect density) and performance (development productivity) (see McGarry et al, 2002).

Corporate leaders have encouraged each organizational unit to develop their own version of the standard corporate process. The goal has been to balance the 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, and even tasks — are free to "tailor" higher-level processes to their own

needs, but they should be able to document and explain where they have differed from these higher-level processes.

Process and discipline

How constraining are these processes? How much does their discipline impinge on the autonomy of the software professional? In short: a great deal.

CSC, like other larger organizations, uses a hierarchy of increasingly fine-grained forms of process discipline:

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

The "granularity" of process discipline at its finest levels -- the degree of formalization and standardization -- can be gauged by the Work Instructions at Program C. There are separate Work Instructions that cover high-level design, two types of low-level design, two types of code reviews, one for testing, and Work Instructions on filling out Change Request Implementation and Resolution forms and Root Cause Analysis forms. Each Instruction is several pages in length. Many include the specific forms to be completed as well as flow-charts detailing the sequence of activities. Overall, the binders presenting the written processes occupy some eight linear-feet of shelf space. In recent years, a growing proportion of this documentation appeared in the form of easier-to-use databases, and more of the procedures specifying work-flows were being built into automated collaboration systems.

If the documentation that developers are required to read is voluminous, so too is the documentation that they are required to write. In the words of one interviewee – perhaps exaggerating for effect:

> "I can write the code in two hours, but then I have to spend two days documenting it. It can be very frustrating. We have to document check-in and check-out, a detail design plan, a development plan. We have to print out all the differences between the old and the new code. There's documentation for inspection and certification. There's an internal software delivery form. A test plan. And all these need to be signed… I used to be an independent developer for about three years. I never even created a flow-chart of my work! The only documentation I needed was a 'to do' list. So I had to change of lot of habits when I got here." (B-2)

To some extent, this formalization is due to size. The typical programs in the four units we studied are large by industry standards – each staffed by upwards of 250 people (see Appendix 2). Moreover, the government clients have imposed significant documentation requirements. And in Programs A and C, further formalization is due to the life-threatening risks associated with the products that the software supports.

HOW DEVELOPERS HAVE RESPONDED

In our interviews, we found two main concerns about process maturity, but these concerns were outweighed by a surprising degree of overall support. A first concern was the burden imposed by greater documentation. Interviewees in all four programs expressed reservations 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-9)

In the two Level 5 organizations, we encountered a second concern, about the validity of the CMM Key Process Areas (KPAs) at Levels 4 and 5. Opinions were divided, but the skeptics expressed themselves in comments such as these:

> "We struggled to get past Level 3. Level 3 seems to give you most of the 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-2)

> "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-4)

Notwithstanding these concerns, the overall reaction of staff at the four units was positive. Program A's case is illustrative. An internal survey of Program A personnel in 1999 asked whether they saw value in the effort associated with certification for CMM. There were 260 responses from 850 surveys distributed. Opinions were largely positive, and even more so for people who had personally participated in an audit. Of those not audited for the CMM, 58% saw CMM as "well worth the effort" and another 30% as "of some value." Of those audited, views were more strongly positive: 79% thought CMM was "well worth the effort" and another 18% thought it was "of some value." The proportion who thought it was of little or no value was 12% among the non-audited and 3% among the audited.

Development staff saw several positive features of CMM and process maturity. We review them in turn, offering in each case illustrative excerpts from our interviews.

Makes for greater collective efficacy

Employees certainly relinquished a great deal of personal autonomy; but the interviews suggested that developers often felt that this individual loss was outweighed by a greater sense of collective efficacy:[1]

> "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: our problem reports. Anyone can write one. The technical leads review them and we track them by category. That allows us to detect 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-14)

> "Perhaps the biggest change as we've become more process mature is that it makes everyone more interested in process improvement. Take an example: now I'm working on a new software utility. Top management asked us to evaluate it, to see if we should all use it. So I've been facilitating a series of meetings with all the managers, where everyone is talking about the utilities they are using and the problems they're having. It's been great to see this kind of problem-solving work going on. That's the effect of having a defined technology change management process. CMM got this process going for us." (D-12)

Helps coordination

Process discipline helped coordinate tasks both within the individual departments and across departments. It helped people see where they fit in:

> "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

[1] On collective efficacy, see Bandura 1997 and Gibson and Early, nd.

> 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-13)
>
> "A well-defined process gives you a kind of map of the whole enterprise." (B-5)
>
> "When the process is documented, I have a clearer understanding of the whole process." (C-2)

Rationalizes paperwork

While overall, the burden of documentation increased with process maturity, this maturity also enabled the organization to be more rational and selective in its documentation requirements:

> "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-9)

Doesn't inhibit creativity

Overall, process did not appear to hinder creativity — or at least not the forms of creativity needed in these programs:

> "Yes, rules can, I suppose, get in the way. But only for the best artists. And I don't know if we want those kinds of people, because their code is going to be very hard to maintain. In our business, we have to ensure that our products can be maintained." (C-15)
>
> "Does Level 5 stifle innovation? Precisely the opposite! Pick a process that makes sense to you, then continually improve it through extensive participation. Why would that inhibit innovation?" (B-13, formerly with Program A)
>
> "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." (B-9, formerly with Program A)

Supports fact-based management

Process maturity appeared to discourage autocratic management styles, as well as the gaming that these styles lead to:

> "With a more mature process, my manager has visibility into how I do my work and can challenge me on it — I can't just play excuses and he can't use brute force." (B-6)
>
> "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-13)
>
> "I think formalized process and metrics can give autocratic managers a club. But it also gives subordinates training and understanding, so it makes the organization less dependent on that manager: he can be replaced more easily. Before I came to CSC, I worked for one of the most autocratic managers you can find. It was always, 'And I want that report on my desk by 5 p.m. today,' with no explanation or rationale. Compared to that kind of situation, an organization with a more mature process leaves a lot less room for a manager to arbitrarily

> dictate how you should work and when work is due. And a more mature process also means that there are more formal review points, so any arbitrary autocratic behavior by a manager will become visible pretty quickly." (D-5)

The CMM as a scaffolding

Education theorists use the metaphor of "scaffolding" to describe the temporary assistance that a teacher provides a student who is trying to master a new task.[2] The CMM functioned as a scaffolding for organizational learning in these CSC units. Even those with doubts about some Level 5 KPAs saw value in others:

> "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-15)

> "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, adding 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 extensive, 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-8)

> "The CMM is helping us move ahead. Just to take an example, even if the CMM didn't exist we would need a technology change management process. Of our 450 people, we have about 50 people in CM, QA, and data management. To move them from one process to another is sometimes like herding cats! The CMM helps us in that by providing an industry-validated approach." (C-10)

The danger, of course, is that the external pressure of CMM certification might encourage a disconnect between the representations made to the outside world and the daily work practices inside:

> "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-13)

SUCCESS FACTORS

In analyzing the reasons for developers' largely positive response to process discipline in these CSC units, we identified four key factors. They form a causal chain, starting with (1) the external impetus to process improvement and CMM certification, leading to (2) the commitment of managers within CSC to pursuing process maturity, to (3) the implementation of a participative process of standardization and formalization, through to (4) a "socialization" of interdependent, collaborative "self-construals" among developers. The four key factors can be arrayed in a table where the organization and the individual are the units of analysis on the two rows, and each unit of analysis has an external context and an internal configuration in the columns:

---

[2] The metaphor of scaffolding, originally articulated by Wood, Bruner and Ross (1976), refers to the temporary assistance provided by teachers/adults to students/children as they strive to accomplish a task in their "zone of proximal development" (Vygotsky, 1962, 1978).

	External context	Internal configuration
Organization level	1. Impetus	2. Commitment
Individual level	3. Participation	4. Socialization

In the following paragraphs, we discuss each of these success factors in turn, identifying the factors that accounted for the broad pattern of success that we found in CMM implementation. In the following section, we will re-visit each of these to identify the associated risks and pitfalls.

Impetus

The driving forces behind CMM-related process improvement efforts at CSC have been twofold: internal goals for process and performance improvement and external legitimacy and credibility. The conjunction of the two impeti seems to have been necessary. Externally, certification at a high CMM level qualified CSC units to bid on government contracts and served as *prima facia* evidence of CSC's superior capabilities. Internally, strong proponents of process discipline in CSC argued that CMM-style process improvement was very worthwhile even absent these external pressures. External pressure would surely have had very different effects had CSC managers believed that the CMM recommendations were technically unsound; but without external pressures, the requisite strong executive-level commitment to process would have been more difficult to mobilize and to sustain.

Clients who supported CSC's process efforts were motivated amongst other things by the visibility and control that high maturity affords them:

> "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-5)

Clients also found CMM certification appealing for its assurance of a faster start-up: practices and infrastructure were already in place. They had access to a documented performance record. They were reassured that the management team put a high priority on quality. And as a result, they felt more confident that program targets would be met (see also Gansler, 2000; Etter, 1999; Aldridge and Stenbit, 2003).

Commitment

Impetus encouraged, but did not guarantee, a corresponding management commitment. Process maturity required a substantial commitment of senior management time and attention and of organizational resources. This was especially true if CMM ratings efforts were going to serve as a means of improving the real process of software development, as distinct from merely developing a facade of documentation that employees did not actually use in their work.

Examining CSC programs as they have moved from low to high maturity levels, several features of this management commitment stand out. At high maturity levels, process — as distinct from program status — was routinely addressed in management review meetings. Management at all levels participated in process related activities, establishing guidance for subordinates concerning the appropriate use of process, actively setting process goals, and reviewing the program's progress towards these goals. Management commitment was visible in the organization's aggressive process programs. This strong commitment was reflected in incentive policies that signaled the importance of process, in resource allocation to process-related activities, and in a reluctance to approve "waivers" from involvement in improvement or assessment activities.

A hallmark of the high-maturity programs at CSC was the commitment of organizational resources. Specialized Process Engineering, Quality Assurance, and Configuration Management staffs were regularly funded. In practice, Program A devoted some 1.75% of its total headcount to quality assurance activities and another 1.25% to process improvement activities. Program C had similar levels of commitment with 1.25% of the total headcount devoted to process improvement, and about 3.25% devoted to quality management. More subtly, the high-maturity programs ensured that the personnel assigned to these functions were highly competent and credible.

Participation

Given external impetus and management commitment, the key to ensuring developers' positive response to process discipline was extensive participation. The high-maturity CSC programs had both formalized processes supporting participation and strong normative commitments to participative rather than autocratic styles of leadership. On Program C, for example, 19.5% of the total headcount were actively involved at some time in the course of the year in either the Software Engineering Process Group (SEPG) or its subordinate process action teams. All organizational components were represented and participated in the process change management process. In the words of one senior program manager:

> "Executive management might impose on me as program manager a CMM rating goal without much dialogue, and they might even have a pretty coercive view of what process discipline consists of. But I can't let that flow downhill from me. We explain to our managers why the rating is important to our future. And once you see the heads nodding, then you have to find the right people for the implementation team — people who aren't going to dictate to the other levels how to proceed. We really can't afford an autocratic style of leadership. The risk of losing critical people is too high." (D-5)

The Process Assurance Cycle. The most advanced form of this combination of discipline and participation was 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.) So the PAC process gives the program manager 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 their group to review. That helps ensure that the S&Ps reflect real-life concerns and that we get buy-in." (A-15)

Software Process Engineering Groups. SEPGs also promoted participation:

> 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. But 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-13)

Developing managers. Given the importance of participation to effective process discipline, it was not surprising to find that executives put great weight on leadership style when developing and selecting managers:

> "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 have such questions, 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-11)

Socialization

Perhaps the most intriguing of our findings is that developers' "self-construals" had changed under the impact of greater process maturity. One's self-construal is the constellation of thoughts and feelings concerning one's relationship to others and the self as distinct from others (Markus and Kitayama, 1991). Traditionally, the self-construal of developers has been a strongly independent one — "hackers" are prototypically low on social needs, with a strong preference for autonomy in their work (Couger and O'Callaghan, 1994). And CSC certainly had its share of people that fit that image. But we were surprised to hear developers describe how the experience of a more mature process had changed these psychological characteristics – their self-construals had shifted from independent hackers to interdependent team-members. They had been "socialized."

This socialization was visible above all in the salience of teamwork and in the value people saw in group efficacy, as distinct from merely individual efficacy:

> "A more mature process means you go from freedom to do things your own way to being critiqued. It means going from chaos to structure. It's a bit like streetball versus NBA basketball. Streetball is roughhousing, showing off. You play for yourself rather than the team, and you do it for the love of the game. In professional basketball, you're part of a team, and you practice a lot together, doing drills and playing practice games. You aren't doing it just for yourself or even just for your team: there are other people involved — managers, lawyers, agents, advertisers. It's a business, not just a game. You have to take responsibility for other people — your team mates — and for mentoring other players coming […] Some programmers here used to be very isolated. We had one fellow who just sa[illegible] his cube all day from six in the morning till two in the afternoon. Many of us didn't even know his name! But the process here drew him into team meetings, and into new conversations. Eventually we even got him helping with training." (B-7)

> "The Improvement Team's work created a real sense of community. Each week it would be someone else's turn to present their process. Since you knew it would be your turn soon, we all helped each other. And everybody got to see how the rest of the organization functioned. All the data was shared. I compare that to the way the organization functioned a few years ago. One of our big problems was poor communications across the organization and up and down the organization. No one knew anything anyone else was doing. Now we're working in unison. Process makes for a more unified front for the customer. And it makes you feel important, because you're part of the process and you know where you're at in the process. I'm only a tiny part of the process, but I know that what I do is needed for the success of the whole thing." (D-12)

This shift from independent to interdependent self-construals appeared to be effected through two mechanisms. First, demographic processes of recruiting, selecting, and differential turnover increased the proportion of personnel who shared the new self-construal (Schneider, 1987):

> "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-8)

Second, the nature of the work encouraged the emergence of interdependent self-construals among the people already there:

> "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-13)

> "I was not originally a believer in this process stuff. I remember seeing coding guidelines when I joined Program D. I just threw them into a corner. But a year later, I found that my code didn't make it through the code checker, and that got me to reconsider. So I went to some CMM training a few years ago. And I've been converted! Most of the developers and leads are being dragged into process kicking and screaming. Any coder would rather just hack. And we have a lot of people who've never worked on a large program and have never gone through the nightmare of losing the source code." (D-22)

PITFALLS AND COUNTER-MEASURES

Following the same causal chain we have followed for success factors, we can trace some of the corresponding pitfalls — the risks, issues, and constraints that can limit progress towards effective pursuit of process maturity — and some of the counter-measures that these CSC programs have found effective.

Weak impetus

In some programs, external pressures had not been consistent. Some customers' support and recognition of the importance of process had oscillated with changes in their own organizations (Basili, McGarry, Pajerski and Zelkowitz, 2002). Under these conditions, it was difficult if not impossible to build process maturity:

> "The biggest problem here has been the customer and getting their buy-in. At Program A, our customer grew towards process maturity with us. Here [at Program B], we started with a less mature client. Some of the customer management even told us that they didn't want to hear about QA or our quality management system — they saw it as wasteful overhead. When you bid a program, you specify a budget for QA and so forth, but if they don't want to pay, you have a resource problem. And once you get the contract, then you start dealing with specific program managers within the customer organization, and these managers don't necessarily all believe in QA or simply don't want to pay for it out of their part of the budget. On the Y2K program, the customer kept changing standards and deadlines. Basically, we were dealing with a pretty process-immature customer. Things have improved considerably since then." (B-13)

> "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-14)

Faced with such challenges, the experience at CSC suggests that development organizations can be proactive in creating greater process commitment in customer organizations. Some programs

found it useful to invite customer representatives into weekly review meetings. More often, the best way to mobilize customers to help in assuring process discipline was to translate process issues into "bottom line" indicators:

> "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-5)

The difficulty, of course, is that it is only at high maturity levels that organizations are able to quantify these costs of immaturity.

Wavering commitment

Process efforts were easily undermined when internal management commitment wavered (Verner, Overmyer and McCain, 1999). Under the pressures of schedule and profitability, funding for the requisite specialized staff was not always available, and management was not always willing to live by the discipline of process. Sometimes senior management's goals in process improvement efforts gave disproportionate weight to the symbolic benefits of certifying formal procedures relative to the technical benefits of rationalizing real work practices. At the project level, the perception was that senior management was largely focused on the bottom line and earnings per share.

> "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-14)
>
> "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-8)

Staff sometimes reacted with frustration where they felt management had not made serious enough efforts to support, streamline and automate the process and thus alleviate its burden.

> "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-3)
>
> "All these forms have a valid purpose, but it takes so long to fill them out that it just doesn't seem very efficient. We really need a lot more automation in doing all this." (B-6)
>
> "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-5)
>
> "The key issue moving forward, I think, is that we still don't have the resources we need to devote to process. A program of this size should have a full-time staff dedicated to our internal process maintenance." (C-7)

The challenge here is the one that executives have faced in other industries and other epochs. When Frederick Taylor developed his revolutionary Scientific Management techniques for bringing discipline to the manufacturing shopfloor, the greatest impediment he encountered was not union opposition but management's reluctance to invest in the requisite "overhead" staff activities (Nelson, 1980).

The counter-measure to wavering commitment was a strong and subtle effort by process champions to convince higher-level executives that these activities are investments, not expenditures. Having gone as far as CMM Level 3, further investment in process can yield big returns by

streamlining and automating the process to alleviate the documentation burden, and tailoring it to further reduce the burden for small-scale, low-risk tasks (Diaz and King, 2002). Just as Taylor found that a well-staffed planning department was needed to revolutionize the factory shopfloor, CSC has found that a dedicated support staff is needed to revolutionize software development.

Insufficient participation

Participation was sometimes less than optimal. Interviewees mentioned instances of insufficient effort by some managers to explain the rationale of some process requirements or to involve staff in process definition, and unresponsiveness to bottom-up suggestions for improvement. Management did not always "listen," or if they listened, did not always "hear":

> "How managers react depends a bit on how you present your suggestion. If you present your idea constructively, they're more likely to react positively. If you come at them with complaints and negative criticism, they don't take it as well. Managers are people too! And sometimes how receptive they are depends on other things going on. For example, if they are under pressure from their bosses to move faster, they may not be very receptive to taking time out to redefine the process." (B-14)

In the more mature programs studied, lower-level staff often participated in tailoring; in contrast, such participation was significantly less developed in the less mature programs. The Level 5 Programs had "improvement teams" in every subunit; in contrast, such teams were only occasionally operative in the Level 3 programs. In general, at Level 5 compared to Level 3 there was both more discipline and more participation at all levels of the organization.

One factor limiting participation was managers' excessive caution about burdening program staff with process-related responsibilities. Often at Level 3, but also sometimes at Level 5, process work was seen as the responsibility of specialized staffs and off-line improvement/action teams. The benefits of such a division of labor are obvious. The downside — the opportunity cost — is that process improvement was not seen as one of the program organization's core tasks, and unless the program organization sees process improvement as one of its own concerns, it is unlikely to collaborate whole-heartedly with process improvement efforts: these latter will appear as intrusive disruptions. When the CMM-related work of process improvement teams appeared to program managers and staff as a costly distraction from their core task, the gap between the formal process and actual practice was likely to be larger, and efforts to narrow were likely to be slowed.

The counter-measure to insufficient participation was to measure, manage, and reward participation more actively. As "software factories," high-maturity development organizations might usefully adopt advanced manufacturing's model of a Learning Factory — where each and every organizational member is actively engaged in organizational learning and improvement efforts.[3]

Resistance to Socialization

Where external impetus and executive commitment were present, where management pursued participation policies, and where process maturity was greater, the (relatively infrequent) reservations about process discipline were primarily from people who appeared as either indifferent or simply too entrenched in their individualistic orientation:

> "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

---

[3] See Adler (1993), Roth, Marucheck, Kemp and Trimble (1994), Leonard-Barton (1992), and Spear and Bowen (1999). These articles describe the involvement of shop-floor production personnel — as distinct from, but collaborating with, dedicated process engineering staff — in learning/improvement activities. It is notable that in these studies, we see considerable employee involvement in the context of highly routine activities — activities where theory predicts that we would be far less likely to see production personnel involved in learning/improvement activities than in the case of software development.

> 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-14)

The counter-measure here was to devote more attention to the "soft" sides of process (culture, leadership, strategic vision) as distinct from its "hard" side (tools, systems, procedures). Vision helps: management needs to offer a credible, positive vision of software process maturity; the image of a Learning Factory mentioned in the previous paragraph is one candidate. Managers also needed to ensure that new recruits benefit from the experience and mentorship of a mix of program work and improvement activities. A recent culture assessment of the culture changes in Program C summarized the key elements of the high-maturity culture shift in these terms (Irion-Talbot, 2002):

- Everyone's world-view becomes less parochial and stove-piped and more enterprise-wide;
- Instead of a focus on their own individual tasks and deliverables, people think of work in terms of interconnected processes that can be statistically characterized;
- Change tolerance increases, indeed change is embraced;
- Leadership is no longer merely a matter of setting goals, allocating resources, and monitoring progress — leadership action becomes more personal, actively engaged, creating and communicating vision; and
- The heroes of the organization are no longer the fire-fighters but the facilitating leaders.

* * *

Managers of software development organizations are understandably cautious about process discipline, fearing that moves in this direction might adversely affect retention of scarce, talented staff. But if our findings at CSC are representative, the great majority of developers will find process maturity empowering, as long as it is implemented in a highly participative fashion. However, moving towards greater process discipline when staff is both suspicious and mobile represents a considerable management challenge. It requires leadership that is simultaneously strong and participative.

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):

The Capability Maturity Model for Software

Level	Focus and description	Key Process Areas
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.	
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
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
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
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

Level 1 characterizes an organization relying exclusively on individual skill and effort, lacking any process supports. Level 2 introduces process for managing individual programs. Level 3 addresses the management of the organization's portfolio of programs. Levels 4 and 5 address how the organization quantifies and improves its processes over time. The underlying philosophy of this hierarchy was inspired by Crosby's (1979) five stages of TQM maturity (uncertainty, awakening, enlightenment, wisdom, certainty) (see Humphrey, forthcoming).

Evidence has accumulated that moving up the CMM hierarchy leads to improvements in product cost, quality, and timeliness. At Lockheed Martin's space shuttle unit in Houston, for example, over a five-year period of sustained process improvement effort leading up to a Level 5 rating, defects were reduced by 90%, cycle time by 40%, and development costs by 40% (Gibbs, 1994, p. 86). (Other case studies of Level 4 and 5 organizations include: Crosstalk, 1999; Dutta and Van Wassenhove, 1998; Humphrey, Snyder, and Willis, 1991; Diaz and Sligo, 1997; Pitterman, 2000; Keeni, 2000; Wigle and Yamamura, 1999; McGarry, et al, 2002, Butler and Lipke, 2000; Willis et al, 1998; see also the reports on the SEI "High-maturity workshops" conducted in 1999, 2000, 2001 available at www.sei.cmu.edu.)

Several multi-organization studies also lend support to the thesis that improved process discipline as reflected in higher CMM levels results in improved productivity. According to one such study (Clark, 1999), total development costs decreased by 5% to 10% (depending on size) for every further level attained. Another study (Harter, Krishnan, Slaughter, 2000) examined 30 software programs in the systems integration division of a large IT firm over the period 1984-96, and estimated the effects of moving from Level 1 to 2 to 3 to be of a similar magnitude for effort, and even greater magnitude for quality and cycle time. (Other multi-organization studies include Krishnan, Kriebel, Kekre, Mukhopadhyay, 2000; Herbsleb, Zubrow, Goldenson, Hayes, Paulk, 1997.)

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. The choice of these programs was guided by a research strategy that sought to identify the effects of "high maturity" processes and to compare the nature of work under these conditions with work performed under somewhat less-disciplined processes. Interviews were conducted a total of 68 people at varying levels of authority and in the main departments in each program. Interviews lasted approximately one hour. They were tape-recorded and interviewees were assured anonymity. The recordings were transcribed, and edited versions were sent back to interviewees for review and correction.

Adler replaced interviewees' names with identification numbers, and organized the interview materials under a logical structure. Working together, all four authors jointly developed the interpretation offered in this paper.

APPENDIX 3: BACKGROUND ON THE FOUR PROGRAMS STUDIED

Program A: CMM Level 5

Program A has had a continuous contractual relationship with its customer for 30 years. Many employees have been attracted to Program A because of the high public profile of the customer. Historically, Program A has about 20 programs under way at any one time, each building and maintaining mid-sized (100-400,000 lines of source-code) subsystems for the complex infrastructure required by the customer. Program A has relied mainly on established technology; however, over recent years, the program's tasks have become more complex as the customer requirements and the associated technologies have evolved. The business environment has also become more demanding, with considerable pressure for more code reuse and tighter deadlines.

Unlike the other three programs, direct customer pressure was not the proximate cause of Program A's commitment to the CMM. The program's management saw the adoption of the CMM as an opportunity to improve their development process and as way to add credibility for potential future customers. In 1991, the first formal, external Software Capability Evaluation (SCE) rated the organization a Level 1. The organization subsequently undertook several internal self-assessments. In 1996, the second evaluation rated them close to Level 3. In 1998, they were assessed as a Level 5 organization.

Over the last 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 B: CMM Level 3

Program B's mission is to build information resource management tools for its government client to use in operations around the world: internal accounting, management, IS resource management, and so on. Program B's staff develop new systems, maintain and upgrade old ones, and operate a Help Desk.

CSC won the contract in 1998 by promising to leverage CSC's experience in Program A to help Program B reach CMM Level 3 within 18 months. CSC replaced nearly 30 contractor organizations who worked largely independently of each other. Program B itself employs directly or indirectly about 300 people. The largest of its sub-programs employs about 90 people building and maintaining a system of some 700 files, comprising about 1 MSLOC.

To help reach Level 3, several people were transferred from Program A. The two largest programs were each led by former Program A people, and Program A veterans staffed several other key management and staff positions. These people used Program A's process as a starting point, and Program B managers were mobilized to tailor this process to their requirements.

Program B's process effort were slowed down by a very difficult Y2K program, which strained relations with the client. That completed, relations improved, and the program was officially assessed as Level 3 in early 2000.

Program C: CMM Level 5

This program, like Program A, has had a continuous contractual relationship with its DoD end-customer for some 30 years. But the relationship had always been mediated by other organizations serving as prime contractors.

Program C employs some 400 technical staff. It undertakes two to five major programs at time, each representing about 2.5-3.2 million source lines of code (MSLOC). These programs create new versions of the weapons control systems they provide to the DoD. Internally, Program C is

divided into four main units that develop and maintain the main modules of the system, plus several support departments.

The key drivers of process maturity at Program C have been the succession of Military Standards imposed by the end-customer (the government) in conjunction with the intermittent pressure of their immediate customer (the prime contractor). By the middle of 1998, Program C was evaluated at Level 4, with all but some minor elements of Level 5 in place as well. In 2001, it was evaluated at Level 5, and has maintained this rating in subsequent evaluations. The quality of its products was widely recognized: the program customer satisfaction index consistently averaged over 97%.

Program D: CMM Level 3

Program D developed infrastructure systems for the DoD. CSC developed its proposal in 1987 and the contract opened in 1991. It had developed 2 million lines of operational code over the 1993-1999 period. Program D is unusual within CSC because it covers the whole product lifecycle, offering complete solutions including hardware, the integration of hardware and software, warehousing, installation, and on-going support. Program D is also unusual within CSC for its extensive use of COTS hardware and software. Its systems incorporate over 200 commercial products. Its systems are being used in about 100 sites, of which about 50 are inter-linked. In 2001, Program D employed some 350 people directly, plus a further 120 contractors.

Traditionally, software process has received less attention in this program than in the other three we studied. However, as part of a bid for very large DoD contract, Program D had to undergo an external process evaluation. In preparation for that evaluation, they conducted their own assessment, and discovered that the program would likely be rated no higher than a Level 1. As a result, the general manager chartered an Improvement Team and charged it with taking the program to Level 3. QA was significantly strengthened — the staff grew from three to eight people — and a broad effort at process documentation was undertaken throughout the organization by department-level Action Teams. By the end of the 1999, the program was assessed as Level 3.

REFERENCES

Adler, P.S., "The Learning Bureaucracy: New United Motors Manufacturing, Inc." in Barry M. Staw and Larry L. Cummings (eds.) Research in Organizational Behavior, vol. 15, 1993, pp. 111-194, Greenwich, CT: JAI Press.

Aldridge and Stenbit, 2003, Memorandum for Secretaries: http://www.acq.osd.mil/SIS/Publications/804memo1.pdf

Bandura, A. 1997, Self-efficacy: The exercise of control, New York: W.H. Freeman

Basili, McGarry, Pajerski, Zelkowitz, "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

Butler, D.L., 1998, "In search of the architecture of learning: A commentary on scaffolding as a metaphor for instructional interactions," Journal of Learning Disabilities, 31, 4, July 1998: 374-385

Butler, K., and Lipke, W., "Software process achievement at Tinker Air Force Base, Oklahoma," CMU/SEI-2000-TR-014.

Butler, T., Standley, V., and Sullivan, E., "Software configuation management: A discipline with added value," Crosstalk – The Journal of Defense Software Engineering, 14, 7, July 2001, 4-8

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

Conradi, Rediar, and Alfonso Fuggetta, 2002, "Improving software process improvement," IEEE Software, July/August, 92-99

Crocca, William T., 1992, review of "Japan's Software Factories: A Challenge to U.S. Management," Administrative Science Quarterly, 37, 4: 670-674

Crosby, P. B., Quality is Free, New York: McGraw-Hill, 1979

Crosstalk, Special issue on CMM Level 5 at the Ogden Air Logistics Center, May 1999, 12, 5

Couger, J.D., and R. O'Callaghan, 1994, "Comparing the motivation of Spanish and Finnish computer personnel with those of the United States," European Journal of Information Systems, 3, 4, 285-301.

DeMarco, T., and Lister, T., Peopleware: Productive Programs and Teams, New York: Dorset, 1987

Diaz, M., and King, J., "How CMM Impacts Quality, Productivity, Rework, and the Bottom Line", Crosstalk, March 2002

Diaz, M., and Sligo, J., "How software process improvement helped Motorola," IEEE Software, 14, 5, Sept-Oct 1997: 75-81

Dutta, S. and Van Wassenhove, L., "Motorola India and Excellence: Life beyond CMM Level 5," INSEAD case study, 1998

Etter, Dr. Delores, "Acquisition Software Oversight," Crosstalk Aug., 1999

Gansler, J.S., "Memorandum for Component Acquisition Executives," Crosstalk Jan. 2000

Gibbs, G.G., "Software's chronic crisis," Scientific American Sept. 1994, p. 86

Gibson, C.D. and P. C. Earley, n.d., "Work-team performance motivated by collective thought: The structure and function of group efficacy," unpublished, University of Southern California.

Harter, D.E., Krishnan, M.S., Slaughter, S. A., "Process maturity effects in software development," 46, 4, April 2000, pp. 451-466

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

Humphrey, Watts S., "Three process perspectives: Organizations, Teams, and People," forthcoming in ACM Annals of Software Engineering

Humphrey, Watts S., Snyder, T.R., and Willis, R.R., "Software process improvement at Hughes Aircraft," IEEE Software, 8, 4, July 1991, 11-23

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

Keeni, G., "The evolution of quality processes at Tata Consultancy Services," IEEE Software, July-Aug 2000: 79-88

Krishnan, M.S., Kriebel, C.H., Kekre, S., Mukhopadhyay, T., "Productivity and quality in software products," Management Science, 46, 6, June 2000, pp. 745-759

Leonard-Barton, D. A. "The Factory as a Learning Laboratory." *Sloan Management Review* 34, no. 1 (1992): 23-28.

Lieberman, H. and Fry, C., 2001. "Will software ever work?" Communications of the ACM, 44, 3: 122-124.

Lynn, Leonard H., 1991, "Assembly line software development," Sloan Management Review, 88:

Markus, H.R., and Kitayama, S., 1991, "Culture and the self: implications for cognition, emotion, and motivation," Psychological Review, 98, 224-253

McGarry, Frank and Decker, William, "Attaining Level 5 in CMM Process Maturity", IEEE Software, November/December 2002: 87-96

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

Pitterman, B., "Telecordia Technologies: The journey to high maturity," IEEE Software, July-Aug. 2000: 89-96

Roth, Aleda V., Marucheck, Ann S., Kemp, Alex, Trimble, Doug, 1994, "The knowledge factory for accelerated learning practices, "Planning Review; Dayton; May/June 1994

Schneider, B. "The people make the place," Personnel Psychology, 1987, 40, 437- 454.

SEI, "Process Maturity Profile of the Software Community 1998 Update", SEMA 12.98

SEI, "Process Maturity Profile of the Software Community, 2002 Mid-year Update." Download from http://www.sei.cmu.edu

Spear, S. and Bowen, H.K., "Decoding the DNA of the Toyota Production System," Harvard Business Review, Sept.-Oct. 1999, 970-106

Standish Group Chaos study report at www.standishgroup.com

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

Vygotsky, L.S., 1962, Thought and Language, Cambridge MA: MIT Press

Vygotsky, L.S., 1978, Mind in Society, Cambridge MA: Harvard University Press

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

Willis, R.R., Rova, R.M, Scott, M.D., Johnson, M.I., Moon, J.A., Shumate, K.C., Winfield, T.O., Hughes Aircraft's widespread deployment of a continuously improving software process, SEI Technical Report CMU-SEI-98-TR-006.

Wood, D., Bruner, J., and Ross, G., 1976, “The role of tutoring in problem-solving,” Journal of Child Psychiatry and Psychology, 17, 89-100