Software Process Improvement: Ten Traps to Avoid[1]

Karl E. Wiegers

Process Impact
716-377-5110
www.processimpact.com

Surviving in the increasingly competitive software business requires more than hiring smart, knowledgeable engineers and buying the latest development tools. You also need to use effective software development processes, so those smart engineers can systematically use the best technical and managerial practices to successfully complete their projects. More organizations are looking at software process improvement as a way to improve the quality, productivity, and predictability of their software development, acquisition, and maintenance efforts. However, software process improvement efforts can be derailed in many ways, leaving the members of the organization jaded, frustrated, and more committed than ever to the ways of the past.

This paper describes ten common traps that can undermine a software process improvement program. Learning about these process improvement killers—and their symptoms and solutions—will help you prevent them from bringing your initiative to its knees. However, it is important to realize that none of the solutions presented here are likely to be helpful if you are dealing with unreasonable people.

Objectives of Software Process Improvement

As you design, implement, and adjust your software process improvement program, it's important to keep these four primary objectives of process improvement in sight:

1. To understand the current state of software engineering and management practice in an organization.
2. To select improvement areas where changes can yield the greatest long-term benefits.
3. To focus on adding value to the business, not on achieving someone's notion of "Process Utopia."
4. To prosper by combining effective processes with skilled, motivated, and creative people.

It is easy to lose sight of these objectives and become distracted by the details of the process improvement model you are following. Emphasize the business and technical results you are trying to achieve, and avoid the checklist or audit mentality that seems to provide a simple, tempting approach.

---

[1] This paper was originally published in *Software Development*, May 1996. It is reprinted (with modifications) with permission from *Software Development* magazine.

Trap #1: Lack of Management Commitment

Symptoms: While individual groups can improve the way they do their work through grass roots efforts, sustainable changes across an organization require management commitment at all levels. Senior managers may claim to support process improvements (how can they say otherwise?), but they may not really be willing to make short-term sacrifices to free up the resources required for the long-term investment. Larger organizations must establish alignment between senior management and one or more layers of mid-managers.

If you're leading the software process improvement effort, you might obtain senior management commitment, but get pushback from middle managers. In this case you'll be forced to spend time and energy debating the importance of software process improvement with people who should only have to be educated, not sold.

Such mixed signals from management make it hard for team leaders and software developers to take the effort seriously. Watch out for lip service and buzzwords masquerading as commitments. Lower-level managers who assign their least capable people (or none at all) to the program are sending a clear sign that the lack of management commitment is about to foil your effort.

Solutions: Managers at all levels need to send consistent signals about software process improvement to their constituencies. Executives must be educated about the costs, benefits, and risks so they will have a common vision and understanding of this complex area of software development. Commitments need to be aligned along the organizational hierarchy, so that managers are not working at cross purposes, and a reluctant manager cannot sabotage the effort through inaction. Make sure that commitments from management translate into resources devoted to software process improvement, realistically defined expectations of the process group and the engineering staff, and accountability for the results.

Management commitment to software process improvement also affects the morale and dedication of people who are working to advance the cause of better processes in the organization. When management objectives change with the wind and the staff devoted to facilitating process improvement is downsized, those affected may be embittered at having months or years of their technical careers sidetracked for nothing. Once burned in such a fashion, they may be reluctant to step forward the next time the organization is looking for people to enable change.

Trap #2: Unrealistic Management Expectations

Symptoms: Excessive enthusiasm by ambitious managers can also pose risks to the improvement program. If the goals, target dates, and results expected by managers are not realistic, the software process improvement effort is ultimately set up for failure. Managers, particularly those with little software experience, may not appreciate the effort and time involved in a large-scale software process improvement effort, such as one based on the Software Engineering Institute's five-level Capability Maturity Model[SM] (CMM[SM]). These managers may be confused about how process improvement frameworks like the CMM relate to other software engineering approaches, such as a specific object-oriented methodology. They may focus on issues of pressing importance

---

[SM] Capability Maturity Model and CMM are service marks of Carnegie Mellon Institute.

to them that are not realistic outcomes of the process improvement effort. For example, a manager may hope to solve current staff shortages by driving the organization to reach CMM Level 2, which typically leads to higher software productivity and quality. However, since it can take two years or more to reach Level 2, this is not an effective solution to near-term staffing problems.

Management needs to understand that the behavioral changes and organizational infrastructure that are parts of a successful software process improvement program cannot be mandated or purchased. Catchy slogans like “Level 5 by ‘95” or “Six Sigma by ‘96” are not constructive. In an unhealthy competitive environment, process improvement can become a contest: Department A sets an objective of achieving CMM Level 3 by the end of 1997, so the head of Department B says that they can do it by the *middle* of 1997. With rare exceptions, such behavior is neither inspiring nor motivating.

Solutions: Educate your managers to help them to understand the realities of what a serious process improvement initiative will cost and what benefits they might expect. Collect data from the software literature on results that have been achieved by other companies with effective improvement programs and the investments those companies made over a specified time period. Every organization is different, so it is risky to promise an eight-fold return from each dollar invested just because you read that some company actually achieved that level of success. Use data available from the software literature or from other areas of your own company to help your managers develop realistic expectations and set reasonable, even ambitious, goals. Software process improvement is no more of a magic silver bullet than any other single software tool or technology.

Trap #3: Time-Stingy Project Leaders

Symptoms: When a senior manager states that he or she is committed to improving the software processes used in the organizations, most project leaders will say that they are, too—whether they mean it or not. However, successful software process improvement initiatives require project leaders to adjust their project schedules to permit team members to devote some time to improvement activities. A project leader who claims to believe in software process improvement but who treats it as a burden added on top of the project activities is sending conflicting signals.

Even if team members are permitted to work on improvement tasks, these tasks often get low priority, and “real work” can easily squeeze process improvement activities out of a busy engineer’s schedule. Project leaders may respond to the pressure of delivering the current product by curtailing the effort that should go into upgrading the organization's process capability.

Solutions: You need to have consistent, active commitment at *all* stages of management; a bottleneck anywhere in the organizational hierarchy can bring the software process program to a screeching halt. One way to achieve consistency is through an interlocking management commitment process as a corporate or organizational policy. Top managers publicly state their goals and priorities (including software process improvement), and people at the lower management levels write their goals and priorities to support those of their superiors.

Senior management must make it clear that project leaders will be evaluated on the effectiveness of their process improvement activities, as well as on the success of the software projects themselves. Software project planning needs to account for the

staff resources that are being devoted to design and implement the new software processes. In small organizations with shallow management hierarchies, the first-level manager is the most critical factor in the success of any process improvement effort. If this person doesn't make software process improvement a visible priority, it just isn't going to happen.

One way to keep a program viable is to treat all software process improvement activities as mini-projects, to give them the visibility and legitimacy they need for success. Write an action plan for each mini-project. This plan identifies resources, states timelines, itemizes deliverables, clarifies accountability, and defines techniques to assess the effectiveness of new processes implemented as a result of each mini-project.

The need to treat improvement activities with the respect afforded to technical projects does not require extensive documentation; most action plans can be written in just one or two pages. Don’t try to solve every process problem in your group at once. Instead, concentrate on the two or three top priority items, as determined through some process assessment mechanism, then tackle the next three, and so on down the line.

Project leaders can’t just assign their least effective people to the software process improvement efforts, either. We all want to keep our best people working on technical projects. However, if good people and respected leaders are not active contributors, the process improvement outputs generated will have less credibility with the rest of the organization.

Trap #4: Stalling on Action Plan Implementation

Symptoms: Action plans might be written after a process assessment, but little progress is made on them because management does not make them a clear priority, assign individuals to work on them, or otherwise take them seriously. Managers may never mention the action plans after they are written, so team members get the message that achieving improved processes by implementing the action plans is really not important. The lack of progress on improvement plans is frustrating to those who actually want to see progress made, and it devalues the investment of time and money made in the process assessment itself.

Solutions: As with Trap #3, a good way to turn action plans into actions is to treat improvement activities as mini-projects. Concentrating on just two or three improvement areas at a time avoids overwhelming the project team. You need to measure progress against the plans, and to measure the impact of each action plan on the business results achieved. For example, a plan to improve the effectiveness of unit testing performed by the programmers might include an interim goal to acquire test automation tools and train developers in their use. These interim goals can be tracked easily. The desired business outcome of such an action plan should be a specific quantitative reduction, over some period of time, in the number of defects that slip through the unit testing quality filter.

If your project leaders never seem to make much progress against their action plans, you may need to implement a management oversight function to encourage them to take software process improvement more seriously. For example, in a particular organization I know of, all project leaders must report the status of their action plans every three months, to a multilevel management steering committee. When this occurs, no one wants to be embarrassed by reporting little or no progress on his or her plans.

From one perspective, such periodic reporting reflects appropriate management accountability for the commitments that people have made to improve their software processes. From another, this approach represents a “big stick” strategy for enforcing software process improvement, which is best avoided unless action plans simply are not being implemented. Your culture will determine the most effective techniques for driving action plans to completion. The management oversight approach did achieve the desired effect in the aforementioned organization.

Trap #5: Achieving a CMM Level Becomes the Primary Goal

Symptoms: Organizations that adopt the CMM framework for process improvement risk viewing attainment of a specific CMM maturity level as the goal of the process improvement, rather than as one mechanism to help achieve the organization’s real business goals. Software process improvement energy may be focused on a race to the level N rating, when some energy should perhaps be devoted to other problem areas that can contribute quickly to the quality, productivity, people, and management issues facing the organization.

Sometimes, a company is in such a rush to reach the next maturity level that the recently implemented process improvements have not yet become well established and habitual. In such cases, the organization might actually regress back to the previous maturity level, rather than continue to climb the maturity ladder as it is attempting to do. Such regression is a surefire way to demoralize practitioners who are eager to move steadily toward a superior software engineering culture.

Solutions: In addition to aiming at the next CMM level, make sure your software process improvement effort is aligned with corporate business and technical objectives. Mesh the process improvement activities with any other improvement initiatives that are underway, such as ISO 9001 registration, or with an established software development framework already in use. Recognize that advancing to the next CMM maturity level can take one to three years. It is not feasible to leap from an initial ad hoc development process to a super-sophisticated engineering environment in one fell swoop. Your goal is not to be able to chant, “We’re Level 5! We’re Level 5!” Your goal is to develop improved software processes and more capable development engineers so that your company can prosper by offering higher quality products to your customers more efficiently than before.

Use a combination of measurements to track progress toward the business goals as well as measure the progress of the software process improvement program. Goals can include reducing project cycle times and product defect levels. One way to track software process improvement progress is to perform low-cost interim assessments to check the status of your project teams in various CMM key process areas (such as requirements management, project planning, and software configuration management). Over time, you should observe steady progress toward satisfying both CMM key process area goals and your company’s software success factors. This is the outcome of a well-planned and well-executed program.

Trap #6: Inadequate Training is Provided

Symptoms: A process improvement initiative is at risk if the developers, managers, and process leaders do not have adequate skills and training. Each person involved must understand the general principles of software process improvement, the

CMM and other pertinent software process improvement methodologies, change leadership, software measurement, and related areas.

Inadequate knowledge can lead to false starts, well-intentioned but misdirected efforts, and a lack of apparent progress. This can undermine the improvement effort. Without training, the organization's members will not have a common vocabulary and understanding of how to assess the need for change or how to interpret specialized concepts of the improvement model being followed, such as the CMM or ISO 9001. For example, "software quality assurance" means different things to different people; training is needed to achieve a common understanding of such terms among all participants.

Solutions: Training to support established process improvement frameworks can be obtained from various commercial sources (such as process improvement consultants or training vendors), or you can develop such training yourself. Different participants in the software process improvement activities will need different kinds of training. If you are using a CMM-based approach, the process improvement group members should receive two to three days of training on the CMM. However, four hours of training about software process improvement using the CMM will be enough for most participants.

If you become serious about software process improvement, consider acquiring training in other key software improvement domains: setting up a software engineering process group (SEPG), establishing a metrics program, assessing the process capability of a project team, and action planning. Use commercial sources of training wherever possible. This way you avoid having to create all of your own training materials.

Trap #7: Expecting Defined Procedures to Make People Interchangeable

Symptoms: Managers who have an incomplete understanding of the CMM may expect that having repeatable processes available (CMM Level 2) means that every project can expect to achieve the same results with any set of randomly assembled team members. They may think that the existence of a defined process in the organization makes all software engineers equally effective. They might even believe that working on software process improvement means that they can neglect technical training to enhance the skills of their individual software engineers.

Solutions: Individual programmers have been shown to have a 10-to-1, 20-to-1, or even higher range of performance (quality and productivity) on software projects. Process improvements alone can never equalize such a large range of individual capability. You can close the gap quite a bit by expecting people to follow effective defined processes, rather than using whatever methods they are used to. This will enable people at the lower end of the capability scale to achieve consistently better results than they might get otherwise. However, never underestimate the importance of attracting, nurturing, and rewarding the best software engineers and managers you can find. Aim for software success by creating an environment in which all team members share a commitment to quality and are enabled—through superior processes, appropriate tools, and effective team interactions—to reach their peak performance.

Trap #8: Failing to Scale Formal Processes to Project Size

Symptoms: A small organization can lose the spirit of the CMM (or any other process model) while attempting to apply the model to the letter, introducing excessive documentation and formality that can actually impede project work. This undermines the credibility of software process improvement, as team members look for ways to bypass the official procedures in an attempt to get their work done efficiently. People are reluctant to perform tasks they perceive as adding little value to their project.

Solutions: To reach a specific CMM maturity level, you must demonstrate that your organization is satisfying all of the goals of each key process area defined at that maturity level and at lower levels. The process definitions your group develops should be no more complicated or elaborate than they need to be to satisfy these goals. Nothing in the CMM says that each procedure must be lengthy or documented in excessive detail. Strive for a practical balance between documenting procedures with enough formality to enable repeatable project successes, and having the flexibility to get project work done with the minimum amount of low-value overhead effort.

This non-dogmatic view doesn't mean that smaller organizations and projects cannot benefit from the discipline provided by the CMM. It simply means that the practices recommended by the CMM should be scaled rationally to the size of the project. A 20 hour project should not demand eight hours of project planning just to conform to a CMM-compliant “documented procedure.” Your process improvement action teams should provide a set of scaleable processes that can be applied to the various sizes and types of projects your group undertakes.

Trap #9: Process Improvement Becomes a Game

Symptoms: Yet another way that software process improvement can falter is when the participants pay only lip service to the real objective of improving their processes. It creates the illusion of change while actually sticking with business as usual for the most part. The focus is on making sure a process audit is passed, rather than on really changing the culture of the organization for the better. Software process improvement looks like the current flavor of the month, so group members just wait for this latest fad to pass so they can get back to working in their old familiar ways. Sometimes, the quest for ever-higher process maturity levels becomes a race. Project teams go through the appropriate motions in an attempt to satisfy some aspect of the CMM, but time is not provided for the group to really internalize the corresponding behaviors before management sets its sights on the next CMM maturity level.

Solutions: To succeed with software process improvement, focus on meeting organizational and company objectives with the help of improved software processes. Do not simply try to conform to the expectations of an established framework like the CMM, ISO 9001, or the Malcolm Baldrige quality award. It is not enough to simply create documented procedures to satisfy the letter of some improvement framework; you must also satisfy the spirit of the framework by actually following those procedures in your daily project work.

The CMM talks about *institutionalizing* process improvements, making new practices routine across the organization. Organization members must also *internalize* improved procedures, becoming committed enough to the new processes that they would not consider going back to their old ways of building software. As a process change leader, identify the behaviors you would expect to see throughout the organization in

each improvement area if superior processes are successfully internalized and institutionalized. As a manager, your group members need to understand that you are serious about continually striving to improve the way they build software; the old methods are gone for good. Continuous improvement means just that, not a one-shot game we play so that someone's checklist can be filled in properly.

Trap #10: Process Assessments are Ineffective

Symptoms: If process capability assessments (often led by the SEPG) are conducted without adequate participation by the software development staff, they turn into audits. This can lead to a lack of commitment, buy-in, and ownership of the assessment findings by the project team. Assessment methods that depend solely on the responses to a CMM-based questionnaire can overlook important problem areas that should be addressed. Outside "experts" who purport to identify your group's process weaknesses based on insufficient practitioner input will have little credibility with your technical staff.

Solutions: Process change is a cultural change, and the ease of this cultural change depends on the extent of the team's involvement with the process assessment and action planning activities. Include a free-form discussion with a representative group of project team members as part of the assessment process whenever time permits. This discussion can identify problem areas that might relate to CMM practices that were not covered by an assessment questionnaire, but which can still be profitably addressed. For example, software testing is not addressed by Level 2 of the CMM, but if poor testing practices are hurting a project in a Level 1 organization, you should do something about that soon. Similarly, topics may come up in a project team discussion that are not part of the CMM at all, including management or organizational issues, lack of resources for acquiring tools, and so forth.

Use your SEPG to actively facilitate the change efforts of your project teams, not just to audit their current process status and report a long, depressing list of findings. Identify process liaisons or champions in the project teams to augment the assessment activities of the process group. Those process liaisons can also help drive effective changes into the project team's behaviors and technical practices. The project team must understand that the SEPG is doing software process improvement *with* the members of the project team, not *for* them or *to* them.

Requirements for Effective Software Process Improvement

Successful software process improvement requires the synergistic interaction of several elements. Begin with smart, trained, creative software engineers and managers, who can work together effectively. Consistent management leadership and expectations help grow a culture that shares a focus on quality, with honest appraisal of problem areas, clear improvement goals, and the use of metrics to track progress. Time must be provided for the team members to identify, pilot, and implement improved processes, with every team member becoming involved in the improvement effort over time. Finally, realize that most of software process improvement is based on thoughtful common sense, combined with a commitment to improve the way software engineering is performed in the organization.

As you chart a course to improve your software process capability, be aware of the many minefields lurking below your organization's surface. Your chances of success increase dramatically if you watch for the symptoms that identify these traps as a threat to your software process improvement program and make plans to deal with

them right away. Process improvement is succeeding at many companies. Make yours one of them, by controlling these risks—and others—as well as you can.

Bibliography

Carnegie Mellon University/Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, Mass.: Addison-Wesley, 1995.

DeMarco, Tom, and Timothy Lister. *Peopleware: Productive Projects and Teams.* New York: Dorset House Publishing, 1987.

Maguire, Steve. *Debugging the Development Process*. Redmond, Wash.: Microsoft Press, 1994.

Humphrey, Watts S. *Managing the Software Process*. Reading, Mass.: Addison-Wesley, 1989.

Weinberg, Gerald M. *Quality Software Management, Volume 1: Systems Thinking*. New York: Dorset House Publishing, 1992.

Wiegers, Karl E. *Creating a Software Engineering Culture.* New York: Dorset House Publishing, 1996.