

# Software Process Improvement: Eight Traps to Avoid

Karl E. Wiegers  
*Process Impact*

*Even well-planned software process improvement initiatives can be derailed by one of the many risks that threaten such programs. This article describes eight common traps that can undermine a software process improvement program, their symptoms, and some possible solutions. Stay alert to the threat of these process improvement killers and attack them before they bring your process improvement program to a screeching halt.*

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 apply the best technical and managerial practices to successfully complete their projects. More organizations are looking at software process improvement (SPI) as a way to improve the quality, productivity, and predictability of their software development, acquisition, and maintenance efforts. However, SPI 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 article describes eight common traps that can undermine an SPI 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 is likely to be helpful if you are dealing with unreasonable people. That is a different class of problem.

## Trap No. 1: Lack of Management Commitment

**Symptoms:** Although 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 SPI (how can they say otherwise?), but they may not 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 midmanagers.

If you are leading the SPI effort, you might obtain senior management commitment, but get resistance from middle managers. In this case, you will be forced to spend time and energy debating the importance of SPI 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 buzz words that masquerade as commitments. Lower-level managers who assign their least capable people (or no one) to the program send a clear signal that the lack of management commitment is about to foil your effort.

**Solutions:** Managers at all levels need to send consistent signals about SPI 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 aspect of software engineering. 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.

“Commitment” means more than hearing a manager say, “I’m fully behind this process improvement thing you’re

doing.” Look for this tangible evidence of management commitment:

- Part of the manager’s performance goals or salary depends on success in SPI.
- Adequate resources are provided.
- Managers communicate clear, consistent expectations, and they publicly share the progress that is made.
- SPI leaders have adequate access to key managers to educate them, present issues, share status, and request assistance.
- Managers take effective actions to break down SPI barriers you present to them.
- A reward structure is established for those who seriously pursue and succeed at process improvement.

Management commitment to SPI also affects the morale and dedication of people who work 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 No. 2: Unrealistic Management Expectations

**Symptoms:** Excessive enthusiasm by ambitious managers also can pose risks to the improvement program. If the goals, target dates, and results expected by managers are not realistic, the SPI effort is ultimately set up for failure. Managers, particularly those with little

---

*This article is based on an article originally published in Software Development, May 1996. It is reprinted (with modifications) with permission from Software Development magazine. Capability Maturity Model and CMM are service marks of Carnegie Mellon Institute.*

software experience, may not appreciate the effort and time involved in a large-scale SPI effort, such as one based on the Software Engineering Institute's five-level Capability Maturity Model (CMM) [1]. 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 development method. They may focus on issues of pressing importance 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 can lead to higher software productivity. However, since it can take two years or more to reach Level 2, this is not an effective solution to a near-term staffing problem.

Management needs to understand that the behavioral changes and organizational infrastructure that are parts of a successful SPI program cannot be mandated or purchased. Catchy slogans like "Level 5 by '95" or "Six Sigma by '96" or "9001 by 2001" are not constructive. In an unhealthy competitive environment, process improvement can become a contest: Department A sets an objective to achieve CMM Level 3 by the end of 1998, so the head of Department B says that they can do it by *mid*-1998. With rare exceptions, such behavior is neither inspiring nor motivating.

**Solutions:** Educate your managers to help them 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 period [4-6]. Every organization is different, so it is risky to promise an eightfold return from each dollar invested just because you read that some company achieved that level of success. Use data from the literature or from other areas of your company to help your managers develop realistic expectations and set reasonable, even ambitious, goals. SPI is no more of a magic silver

bullet than any other single software tool or technology.

### Trap No. 3: Time-Stingy Project Leaders

**Symptoms:** When senior managers state that they are 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 SPI 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 SPI 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 SPI 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 SPI), and people at the lower management levels write their goals to support those at the higher levels.

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. Software project planning needs to account for the resources being devoted to design and implement the new software processes. The first-level manager is the most critical factor in the success of any process improvement effort. If this person does not make SPI a visible priority, it is not going to happen.

One way to keep a program viable is to treat all process improvement activities as miniprojects, to give them the visibility and legitimacy they need for success. Write a short action plan for each miniproject. This plan identifies resources, states time lines, itemizes deliverables, clarifies accountability, and defines techniques to assess the effectiveness of new processes implemented as a result of each miniproject. Track the effort devoted to SPI to see if the investment level matches your planned commitment. Do not 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 few, and so on down the line.

Project leaders cannot just assign their least effective people to the improvement efforts, either. If good people and respected leaders are not active contributors, the processes generated will have less credibility with the rest of the organization.

### Trap No. 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 not that important. The lack of progress on improvement plans is frustrating to those who want to see progress made, and it devalues the investment of time and money made in the process assessment.

**Solutions:** As with Trap No. 3, a good way to turn action plans into actions is to treat improvement activities as miniprojects. 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 easily tracked. The desired business outcome of such an action plan should be a specific quantitative reduction, over some period, 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 SPI more seriously. In one organization I know of, all project leaders must report the status of their action plans every three months to a management steering committee. When this occurs, the project leaders do not want to be embarrassed by reporting little or no progress on their 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 to enforce SPI, which is best avoided unless progress is not being made. Your culture will determine the most effective techniques to drive action plans to completion. The management oversight approach did achieve the desired effect in the aforementioned organization.

### Trap No. 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 ultimate goal of the improvements, rather than as one mechanism to help achieve the organization's real business goals. SPI 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 that face the organization.

Sometimes, a company is in such a rush to reach the next maturity level that the recently implemented process changes have not yet become well estab-

lished and habitual. In such cases, the organization might 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 maturity level, make sure your SPI effort is aligned with corporate business and technical objectives. Mesh the process improvement activities with any other improvement initiatives that are under way, 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 supersophisticated 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.

You may be compelled to achieve a specific CMM maturity level by an external driver, such as the need to be able to bid for certain contracts. If you are not so driven, though, adapt the CMM to the shape and needs of your organization and culture to achieve the desired benefits. Do not just aim for the maturity rating because it is a concise, simply stated goal.

Use a combination of measurements to track progress toward the business goals as well as to measure the progress of the SPI program. Goals can include reducing project cycle times and product defect levels. One way to track SPI 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, software project planning, and software configuration management). Over time, you should observe steady progress toward achieving both CMM key process area goals and your company's software success factors.

### Trap No. 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 SPI, the CMM, and other pertinent SPI methods, 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. 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. 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. Different participants in the SPI activities will need different kinds of training. If you are using a CMM-based approach, the process improvement group members should receive several days of training on the CMM. However, four hours of training about SPI using the CMM will be enough for most participants. At Eastman Kodak Co., we developed a series of four-hour overview courses on various software engineering practice areas (requirements engineering and management, peer reviews, project planning and tracking, metrics, and configuration management) for project teams engaged in SPI.

If you become serious about SPI, 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 to avoid having to create all of your own training materials.

### Trap No. 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 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 SPI 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 ratio of 10-to-1, 20-to-1, or even higher range of performance (quality and productivity) on software projects [2, 3]. 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 teammates share a commitment to quality and are enabled—through superior processes, appropriate tools, and effective team interactions—to reach their peak performance.

### Trap No. 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 impede project work. This undermines the credibility of SPI, as

teammates 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 achieve 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 processes you develop 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 extreme 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 nondogmatic view does not mean that smaller organizations and projects cannot benefit from the discipline provided by the CMM. It simply means that the procedures you adopt should be scaled rationally to the size of the project. A 40-hour project should not demand eight hours of project planning just to conform to a CMM-compliant "documented procedure." Your process improvement teams should provide a set of scalable processes that can be applied to the various sizes and types of projects your group undertakes.

### Conclusion

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 SPI program, and when you 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. ♦

### About the Author

**Karl E. Wiegiers** is the principal consultant with Process Impact in Rochester, N.Y. Previously, he spent 18 years at Eastman



Kodak Co., including experience as a photographic research scientist, software developer, software manager, and software process and quality improvement leader. He holds a doctorate in organic chemistry from the University of Illinois. He is a member of the Institute of Electrical and Electronics Engineers (IEEE), IEEE Computer Society, American Society for Quality, and the Association for Computing Machinery. He is the author of the award-winning book *Creating a Software Engineering Culture* (Dorset House, 1996) and has written over 110 articles on many aspects of computing, chemistry, and military history. He is a frequent speaker at software conferences and professional society meetings.

Process Impact  
31 Canterbury Trail  
Fairport, NY 14450-8783  
Voice: 716-377-5110  
Fax: 716-377-5144  
E-mail: kwiegiers@acm.org  
Internet: <http://www.processimpact.com/>

### References

1. Carnegie Mellon University/Software Engineering Institute, *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley, Reading, Mass., 1995.
2. Curtis, Bill, "The Human Element in Software Quality," *Proceedings of the Monterey Conference on Software Quality*, Software Productivity Research, Cambridge, Mass., 1990.
3. DeMarco, Tom and Timothy Lister, *Peopleware: Productive Projects and Teams*, Dorset House Publishing, New York, 1987.
4. Diaz, Michael and Joseph Sligo, "How Software Process Improvement Helped Motorola," *IEEE Software*, September/October 1997.
5. Dion, Raymond, "Process Improvement and the Corporate Balance Sheet," *IEEE Software*, July 1993.
6. Herbsleb, James, Anita Carleton, James Rozum, Jane Siegel, and David Zubrow, "Benefits of CMM-Based Software Process Improvement: Initial Results," Technical Report CMU/SEI-94-TR-13, Software Engineering Institute, Pittsburgh, 1994.